Studenti:

Cassano Lorenzo (matricola 718331)

l.cassano25@studenti.uniba.it

D'Abramo Jacopo (matricola 716484)

j.dabramo@studenti.uniba.it

D'Orsi Domenico (matricola 718938)

d.dorsi1@studenti.uniba.it

RELAZIONE CASO DI STUDIO – CORSO DI INGEGNERIA DELLA CONOSCENZA

A.A. 2021-2022

Strutturazione e creazione di una Base di Conoscenza, clustering e apprendimento di una rete Bayesiana per l'analisi nel dominio alberghiero di Airbnb

Obiettivi del progetto

Gli obiettivi del nostro progetto sono molteplici e ciascuno relativo all'utilizzo che si è fatto dei vari metodi di apprendimento. In generale, abbiamo voluto rendere esplorabile il dataset "*Inside Airbnb*" (http://insideairbnb.com/), del quale abbiamo selezionato, in particolare, i dati relativi alla città di Milano: andando nello specifico

- **Knowledge Base:** abbiamo costruito una base di conoscenza in modo tale da permettere all'utente di verificare, tramite apposite query, svariate proprietà delle stanze, permettendo la verifica sia di proprietà base, come il numero di letti e bagni, sia proprietà più complesse come la possibilità di cucinare
- **Clustering:** abbiamo utilizzato questa tecnica di apprendimento non supervisionato per andare a raggruppare tra di loro camere simili
- **Belief network:** l'apprendimento di questo modello probabilistico ci ha permesso di scoprire correlazioni nascoste tra le varie feature, e calcolare probabilità a posteriori sulla qualità della stanza date delle evidenze (preferenze) fornite dall'utente

Strumenti adottati

Il linguaggio utilizzato per sviluppare il progetto è stato **Python**, data la grande mole di librerie ad agevolezza d'uso delle stesse, in particolare di quelle funzionali ai nostri obiettivi. Le principali librerie da noi utilizzate sono state **Pandas** per la gestione del dataset ed il relativo preprocessing, **PySwip** per poter utilizzare i comandi dell'applicativo **SWI-Prolog** (necessari per la costruzione e l'interrogazione della Knowledge Base), **Sklearn** per la parte relativa al clustering ed infine **Pgmpy** per l'inferenza probabilistica effettuata con la Belief Network. Inoltre, abbiamo anche fatto uso dell'applicativo **Weka** per poter apprendere la struttura della Belief Network partendo dai dati, in quanto non conosciuta a priori

Prima parte: preprocessing dei dati

1. **Cleaning del dataset**
   Il dataset iniziale contiene diverse colonne poco utili al nostro scopo, e per tale motivo la prima operazione consiste nella loro eliminazione, per poi andare a selezionare solo le tipologie principali di proprietà, ed infine andiamo a cancellare tutte le righe che non presentano il valore di "bedrooms". Segue la modifica di alcuni caratteri di varie colonne che potrebbero creare problemi in fase di successivo utilizzo, come eliminazione di spazi e trasformazione di valori 't' in 1 per alcune colonne booleane. E' stato inoltre effettuato il casting di tipo per alcune feature, nonché il riempimento di valori vuoti tramite **Simple Imputer**, utilizzando la media per feature numeriche ed il valore più frequente per quelle categoriche. Infine, viene creata la colonna "is_center" per specificare quali stanze siano collocate in zone centrali della città, con una lista creata ad hoc attraverso una mappa della città di Milano.

2. **Gestione delle "amenities"**
   La colonna originale denominata "amenities" indica tutti i servizi messi a disposizione dalle varie strutture, ma richiede ovviamente di essere modificata per renderla utilizzabile. Dopo una piccola operazione testuale, il primo passo è l'eliminazione delle colonne presenti in un numero di stanze minore di un treshold, che abbiamo fissato al 30% delle strutture totali. Dopo questa operazione, andiamo quindi a costruire la matrice da aggiungere al nostro dataframe.

3. **Preprocessing per la belief network**
   Per la costruzione del dataset utilizzabile, è stato necessario eliminare ulteriori colonne non funzionali all'inferenza probabilistica. Oltre ad alcuni casting ed operazioni testuali, la procedura più importante è quella di discretizzazione di varie colonne, trasformate in valori testuali per rendere più comprensibile e naturale il linguaggio con cui l'utente andrà ad interagire con la belief network: i valori sono stati determinati attraverso l'utilizzo del metodo di inferenza "Variable Elimination", il quale richiamato sui valori non ancora discretizzati va ad indicare gli intervalli per suddividere il dataframe in maniera omogenea, eccetto quella relativa ai prezzi la quale è stata strutturata per

essere coerente con i valori utilizzati per la knowledge base e con i nomi degli intervalli stessi

4. **Preprocessing per il clustering**
   L'ultimo preprocessing necessario è quello per il clustering, in modo tale da produrre un dataset che sia il meglio ottimizzato per questo scopo. Per fare ciò, ci siamo limitati a considerare le principali feature per considerare la similarità tra le camere, come prezzo, numero di recensioni e rating della struttura. Per fare ciò è stato necessario applicare la **Principal component analysis**, per ridurre il numero di features, e uno dei metodi più utilizzati per scalare i valori delle features, ovvero il MinMaxScaler. Dopo aver effettuato il clustering, inoltre, si va anche a definire il dataset che verrà utilizzato per la Knowledge Base, ottenuto semplicemente inserendo all'interno del dataset ottenuto dall'operazione di cleaning una colonna con il relativo cluster per ciascuna stanza.

Seconda parte: clustering

La tecnica del clustering è stata scelta per poter andare ad individuare e raggruppare stanze che presentano similarità tra loro, sfruttando ciò anche per la knowledge base. Questa tecnica, infatti, è atta all'individuazione dei cosiddetti **cluster**, ovvero esempi più simili a dei centroidi calcolati automaticamente.

Il clustering può essere di due tipi:

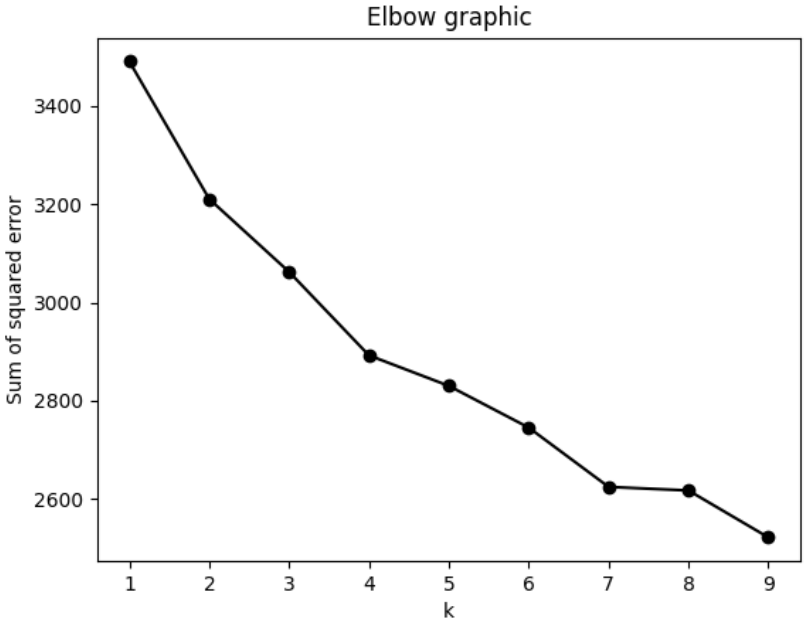
- **Hard clustering,** che prevede un'assegnazione statica di ogni esempio ad una classe di appartenenza
- **Soft clustering**, che adotta delle distribuzioni di probabilità sulle classi associate ad ogni esempio

Nel nostro caso, è stato adottata la prima tipologia: nello specifico, la tecnica da noi utilizzata è quella del **k-means**. Tale algoritmo va ad inizializzare in maniera casuale un dato numero di centroidi (pari al numero di cluster che l'utente ha inserito), per poi fare una prima assegnazione degli esempi del dataset. A questo punto, per ciascun centroide verrà calcolata la media dei valori delle features, i quali verranno utilizzati per il successivo ricalcolo delle assegnazioni: il k-means, infatti, ha come condizione per la convergenza la minimizzazione della sommatoria degli errori quadratici.

$$\sum_{e \in E} \sum_{j=1}^{n} \left(\hat{X}_j(class(e)) - X_j(e) \right)^2$$

Numero di clusters ed iterazioni sono parametri richiesti all'utente, ma nonostante ciò, il numero migliore di clusters è determinato attraverso il **"metodo del gomito"**, il quale permette di individuare il numero di clusters minimo per poter diminuire significativamente l'errore

associato al modello attraverso il plot di un grafico che vada a rappresentare l'errore stesso in relazione ad un crescente numero di clusters



Data l'irregolarità della curva, è difficile individuare univocamente il gomito, ma si può considerare 7 come numero ideale, in quanto si trova subito dopo un cambio significativo di inclinazione, sebbene il numero di cluster sia abbastanza alto.

Andiamo infine a mostrare i valori delle coordinate dei centroidi (media dei valori delle features) dei cluster per le principali features considerate nell'applicazione del k-means, con una computazione effettuata su 7 cluster e 100 iterate

Host response rate	Host is superhost	Is center	Price	Number of reviews	Review scores rating
0.095	0.035	0.097	0.479	0.104	0.373
0.012	0.036	0.096	0.642	0.075	0.251
0.012	0.019	0.782	0.487	0.088	0.381
0.010	0.045	0.097	0.367	0.118	0.373
0.011	0.004	0.157	0.645	0.877	0.306
0.009	0.045	0.099	0.442	0.117	0.400
0.011	0.014	0.098	0.682	0.101	0.593

Come si può notare, sono state considerate solo features numeriche e non categoriche, in quanto il k-means opera con questa tipologia di dati. Inoltre, i valori potrebbero risultare poco significativi, ma in realtà ciò è dovuto all'operazione di scaling precedentemente effettuata.

Terza parte: Belief Network

L'apprendimento e la realizzazione della Belief Network è stato effettuato con lo scopo di evidenziare, attraverso la struttura ottenuta dai dati, il modo in cui le features sono tra loro correlate, e permettere all'utente di eseguire interrogazioni basate sull'inferenza probabilistica.

Una rete bayesiana è un modello che rappresenta in forma grafica una distribuzione di probabilità congiunta su più variabili, alcune delle quali sono dipendenti le une dalle altre. Tale modello, infatti, è rappresentato da un grafo orientato aciclico, il quale indica il modo in cui le features sono tra loro dipendenti, e da una serie di probabilità condizionate. In particolare, la rete rappresenta un ordinamento tra le features in termini di variabili genitori e variabili figlie, ove le prime rappresentano l'insieme minimale dei predecessori di una data variabile figlia, tale che gli altri predecessori di quest'ultima siano condizionatamente indipendenti dalla variabile figlia stessa dati i suoi genitori. La probabilità condizionata sarà quindi:

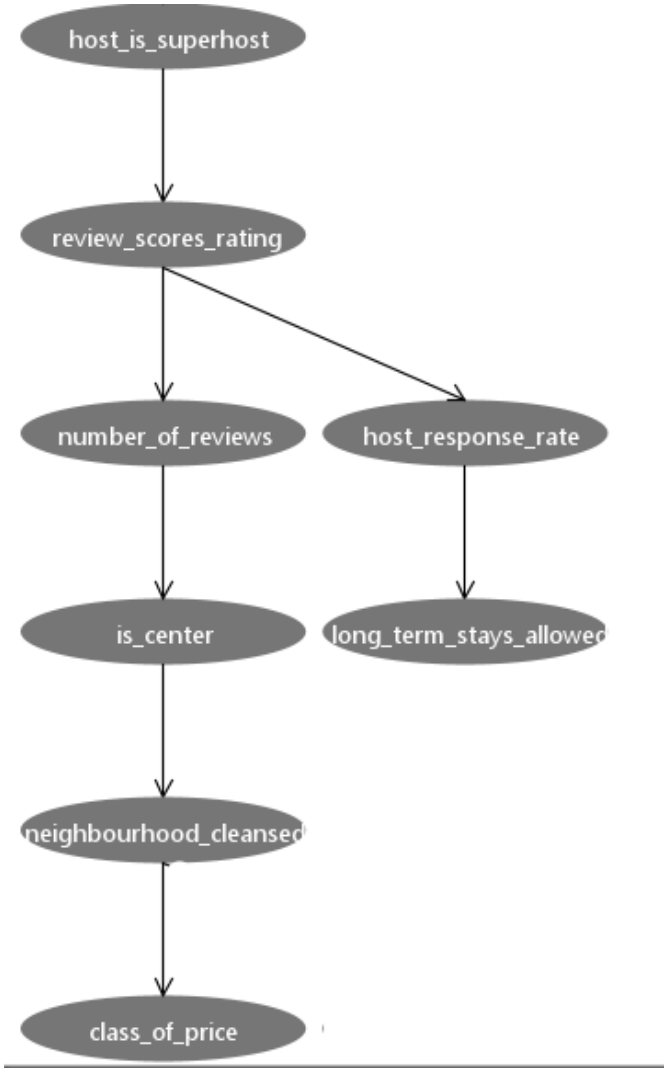
$$P(X_i \mid parents(X_i))$$

Nel nostro caso, però, la struttura non era nota a priori, ed è stato quindi necessario apprenderla attraverso i dati in modo da andare a selezionare il miglior ordinamento possibile. Per fare ciò, è stato necessario utilizzare l'applicativo **Weka**, il quale restituisce la struttura della rete Bayesiana in funzione di alcuni parametri, su cui abbiamo effettuato un accurato testing:

- **Algoritmo di ricerca:** abbiamo selezionato l'algoritmo "Hill Climber", con score type "AIC" (Akaike Information Criterion). Per quanto riguarda il numero di genitori per ciascun nodo, abbiamo voluto lasciare questa scelta all'utente tra tre differenti valori: un genitore, il cui risultato sarà un classificatore **Naive Bayes**, due genitori, il cui risultato sarà una **Tree Augumented Bayes Network,** oppure tre genitori, il cui risultato sarà una **Bayes Net Augumented Bayes Network.**

- **Stimatore:** abbiamo utilizzato il "SimpleEstimator" con α = 0.5, parametro che viene utilizzato per l'apprendimento delle CPT e viene interpretato come conteggio iniziale per ogni valore. Abbiamo deciso di non modificarne il valore di deafult in quanto non ci sono note le distribuzioni dei dati o eventuali pseudo-conteggi.

Abbiamo inoltre ottimizzato il processo di selezione della struttura attraverso la **k-fold Cross Validation**, con un numero di fold pari a 10. Prendendo in esame la struttura ottenuta scegliendo un solo nodo genitore per ciascuna variabile figlia, andiamo a mostrare la parte di struttura più interessante, ed i relativi score ottenuti



Interessante notare come la classe di prezzo dipenda dal quartiere, e come gli host più respondivi siano coloro che permettono lunghe permanenze

LogScore Bayes	-171125.5019531362
LogScore BDeu	-174482.15321895957
LogScore MDL	-173890.53465616005
LogScore ENTROPY	-170899.0085764058
LogScore AIC	-171554.0085764058

Classe di qualità	Precision	Recall	F-Measure
Low Rating	0,624	0,766	0,688
Top Rating	0,557	0,161	0,250
Nice Rating	0,542	0,516	0,529
Good Rating	0,660	0,880	0,754
Media	0,603	0,616	0,580

Classificato come →	Low Rating	Top Rating	Nice Rating	Good Rating
Low Rating	2825	190	351	320
Top Rating	777	320	323	568
Nice Rating	792	1	847	0
Good Rating	130	63	41	1721

Dopo aver appreso la struttura della Belief Network, l'utente può finalmente sottoporgli delle query basate sulle sue preferenze, riguardanti proprietà come il quartiere in cui si trova la struttura, se l'host sia classificato come "superhost" e la presenza di servizi vari. Le preferenze dell'utente saranno poi utilizzate come evidenze per l'algoritmo di Variable Elimination, dove la variabile da predire sarà invece il "review scores rating", ovvero la media delle recensioni rilasciate dagli utenti sulla data struttura. In questo modo andiamo ad indicare, date le preferenze espresse dall'utente, la probabilità con cui le camere presenti nel dataset abbiano un certo livello di qualità, e possano quindi piacergli. La probabilità calcolata sarà quindi *P(review_scores_rating|preferenze utente)*

Query esempio: host_is_superhost = True, is_center = True, class_of_price = medium

Quarta parte: Definizione della base di conoscenza (KB)

Una base di conoscenza (Knowledge Base) in logica di primo ordine è costituita da un insieme di proposizioni, dette assiomi, che sono assunte essere vere senza dimostrazione. La base di conoscenza è utile per rappresentare, all'interno di una macchina, la conoscenza riguardo un particolare mondo. In particolare, la creazione di un kb prevede i seguenti passaggi:

- Decidere il dominio da rappresentare: esso può includere aspetti del mondo reale, un mondo immaginario o un mondo astratto (numeri e insiemi)
- Il progettista deve poi scegliere le proposizioni atomiche per rappresentare il mondo
- In seguito, il progettista deve definire le proposizioni che saranno vere nell'interpretazione intesa (Assiomatizzazione del dominio)
- Infine, si possono porre al sistema delle query, ovvero, determinare se specifiche proposizioni sono conseguenze logiche della KB (proposizione vera in tutti i modelli della KB)

Il sistema, a differenza del progettista, non comprende il significato dei simboli, bensì, è in grado di decidere se una particolare proposizione sia conseguenza logica oppure no, in base agli assiomi presenti nella base di conoscenza. Successivamente il progettista, in base all'interpretazione intesa, comprende se il risultato ottenuto dal sistema è valido oppure no.

Quindi, come ulteriore metodo per la rappresentazione di conoscenza è stato scelto di definire una base di conoscenza in Prolog, ragionando, perciò, in logica di primo ordine. Per poter definire una KB, il dataset è stato prima soggetto ad una fase di preprocessing (descritto nel file cleaning.py) e successivamente si sono scelte le features da assiomatizzare come fatti nella KB. Inoltre, sono state aggiunte alcune regole che permettono all'utente di sottomettere al sistema query più complesse. Per esempio:

- ```
  price_range(Room,Range):-
  price(Room,Range),class(Range,Price).
  ```

  L'utente tramite suddetta query può sapere quali sono le stanze che rispettano un determinato range di prezzo. In particolare, puo inserire 5 classi di prezzo:

  ```
  top_level prezzo>675
  expensive prezzo>200 and prezzo<=675
  medium    prezzo>55 and prezzo<=200
  affordable prezzo>40 and prezzo<=50
  economy prezzo<=40
  ```

- ```
  similar_rooms(X,Y) :- cluster(X,C),cluster(Y,D), C = D.
  ```

  L'utente tramite suddetta query puo verificare se due stanze sono simili sulla base del cluster a loro associato. In particolare, se i cluster delle stanze identificate da `X` e `Y` risultano uguali, allora le stanze saranno simili.

- `connections(X,Y) :- amenities(X,Y), member(Y,[“wifi”,”cable tv”])`

  L’utente tramite suddetta query può verificare se una determinata stanza dispone di servizi come “wifi” e “cable tv”

Di seguito sono riportati esempi di query:

- Esempio semplice di query vera (fatto della KB):

  ```
  neighbourhood_cleansed(32778914,’navigli’).
  true.
  ```

- Esempio semplice di query falsa:

  ```
  host_is_superhost(147525).
  false.
  ```

- Esempio di query non ground:

  ```
  small_rooms(X).
  0 23986
  1 46536
  2 55055
  3 59226
  4 74835
  … …
  7273 53764507
  7274 53808297
  7275 53819768
  7276 53824628
  7277 53830099
  ```

Conclusioni e considerazioni finali:

Come si è potuto constatare dalla documentazione lo scopo principale del progetto è stato quello di analizzare la conoscenza sotto varie forme: la definizione di una base di conoscenza (logica di primo ordine), tecniche di apprendimento supervisionato in forma probabilistica (belief network) e infine tecniche di apprendimento non supervisionato (clustering). L'utilizzo di tali tecniche permette di scoprire pattern interessanti riguardo le diverse strutture registrate su Airbnb nella zona di Milano, per esempio: sottomettere query alla knowledge base per ottenere informazioni circa i servizi, numero letti, bagni, ospiti ed altro, così come verificare la qualità delle camere disponibili all'interno del dataset selezionando le proprietà che si ricercano nella struttura.

Inoltre, la tecnica del clustering (che utilizza l'algoritmo k-means) è stata sfruttata dalla base di conoscenza per poter ottenere informazioni su camere simili.

È importante sottolineare che il sistema può essere ampliato includendo, ad esempio, tecniche di NLP e Path Of Speech Tagging che vadano ad analizzare varie feature delle diverse strutture per individuare la distribuzione di parole chiave all'interno di campi come il nome e la descrizione della struttura, il cui contenuto potrebbe essere altrimenti utilizzato in maniera errata.

Un altro esempio di applicazione derivabile dal nostro studio è l'impiego della conoscenza ricavata attraverso il suo utilizzo per andare a realizzare sistemi di raccomandazione, memorizzando le preferenze e le abitudini dei vari utenti del sistema, in modo da poter anche indirizzare e suggerire alcune camere all'utenza del sistema stesso.

Infine, si fa notare come questo tipo di studi siano estremamente validi, per esempio, per migliorare i servizi offerti da Airbnb: includendo conoscenza che non considera solamente le semplici caratteristiche delle camere bensì anche fattori di "user experience", espandendo e rendendo maggiormente capillare la ricerca da parte dell'utente di una camera che soddisfi le proprie esigenze.