UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

CASO DI STUDIO
INGEGNERIA DELLA CONOSCENZA
TRAVEL ASSISTANT

Gruppo di lavoro

- **Gianpietro Fontana, 736570**, g.fontana10@studenti.uniba.it
- **Simone Lastilla, 717851**, s.lastilla3@studenti.uniba.it

https://github.com/gianpietr0/travelAssistant

AA 2022-2023

Introduzione

Ogni giorno, migliaia di persone viaggiano per scopi differenti, tra cui turismo, lavoro, affari e molto altro. Infatti, il turismo è uno dei settori più grandi al mondo e fondamentali per l'economia di un paese. D'altro canto, l'informatica cerca sempre di semplificare quei compiti quotidiani degli utenti e in questo caso ci sono diversi sistemi a supporto per l'organizzazione dei viaggi o per gli utenti stessi assistendoli per tutta la durata del viaggio. Basta pensare infatti a Google Maps che guida gli utenti verso le loro destinazioni, o Booking che consente la prenotazione di alloggi e molto altro. Talvolta però manca un programma in grado di integrare tutte quelle funzionalità utili relativi ai viaggi, in quanto spesso si specializzano in funzionalità specifiche come la prenotazione di una camera d'albergo o altro.

Nel nostro progetto, si mira nel realizzare un sistema completo in grado di coprire tutti gli aspetti relativi a un viaggio, indipendentemente da quale sia la sua motivazione, in grado di assistere l'utente non solo nella ricerca dell'alloggio che più soddisfa le sue richieste, ma anche nello spostamento all'interno della città in cui si trova. Quindi l'obiettivo è realizzare una sorta di assistente di viaggio a 360 gradi.

La città che viene presa in considerazione per questo sistema è Londra, capitale inglese meta di milioni di turisti o lavoratori.

Per quanto riguarda gli alloggi si considerano gli alloggi offerti da Airbnb. Si tratta si un portale che mette in contatto tra loro persone in ricerca di un alloggio con persone che ne dispongono uno e che desiderano affittarlo per brevi periodi. La particolarità di questo servizio è che gli alloggi non sono necessariamente camere d'hotel, anzi la maggior parte degli host sono privati che affittano singole stanze o intere case.

Per quanto riguarda, invece, la gestione degli spostamenti si va a considerare il sistema di trasporto metropolitano offerto da TFL (Transport for London). Esso è particolarmente come uno dei sistemi più efficienti al mondo in grado di connettere qualsiasi punto nella città rendendolo il mezzo preferito sia dei turisti che dei residenti.

Sommario

Il sistema TravelAssistant è un KBS che ha come obiettivo quello di aiutare i viaggiatori nell'organizzare un viaggio, nell'assisterli durante gli spostamenti e nell'avere una previsione di quanto potrebbe costare un viaggio con determinate condizioni.

Il sistema dispone di una KB costituita da fatti e regole sul dominio considerato, la quale viene interrogata tramite apposite query al fine incrementare il dataset a disposizione, ma anche per consentire il corretto funzionamento delle varie funzionalità fornite dal sistema.

Il sistema procede sia con l'apprendimento supervisionato che quello non supervisionato. Nell'apprendimento supervisionato l'obiettivo è quello di prevedere la classe di prezzo di un certo alloggio. A tale scopo si cerca il modello predittivo e la configurazione di quest'ultimo che più garantisce una maggiore accuratezza predittiva.

Nell'apprendimento non supervisionato, invece, l'obiettivo è quello di raggruppare gli annunci che sono più simili tra loro. Questo viene effettuato mediante il clustering.

Infine, il sistema risolve un problema di ricerca, tramite un apposito algoritmo di ricerca, che consiste nel trovare il percorso ottimale che l'utente deve fare per spostare da una posizione all'altra mediante l'utilizzo della metropolitana. L'ottimalità del percorso è misurata in termini di distanza più breve di quest'ultimo per raggiungere la destinazione.

Le informazioni e i dataset utilizzati nel programma provengono da fonti distinte:

- alloggi (datasetListings) e disponibilità giornaliere di questi (calendar) dal sito Airbnb Open Data, il quale fornisce una serie di dataset sugli alloggi per diverse città internazionali.
- rete metropolitana da London Underground geographic maps/CSV

Elenco argomenti di interesse

- Paragrafo su argomento 1 (es. CSP, rappresentazione della conoscenza: clausole di Horn)
- Paragrafo su argomento 2 (es. Ragionamento automatico)
- Paragrafo su argomento 3 (es. Apprendimento e incertezza, ragionamento su KB distribuite)
- …

(tratti da sezioni diverse del programma, da indicare esplicitamente)

KNOWLEDGE BASE (KB)

Sommario

Una KB, ovvero Knowledge Base, è un insieme di proposizioni dichiarate vere. Ogni elemento prende il nome di assioma. Un modello della KB è un'interpretazione per la quale sia vero ogni assioma della KB. Questa rappresenta tutta la conoscenza circa un determinato dominio [3].
Il tipo di rappresentazione adottata è quella basata su individui e relazioni. Gli individui sono delle entità del mondo rappresentato che possono essere reali o immaginari, astratti o concreti e a cui il sistema può fare riferimento. Le relazioni, invece, rappresentano le verità riguardanti gli individui, ovvero le loro proprietà e associazioni con altri individui [4].
La base di conoscenza è formata da fatti, ovvero verità circa il dominio rappresentato, che costituiscono la conoscenza primitiva, ma anche di regole grazie alle quali è possibile derivare degli ulteriori fatti formando la conoscenza derivata.

Strumenti utilizzati

Per la realizzazione della KB si sono definite proposizioni e clausole tramite il linguaggio Prolog salvandoli in appositi file di tipo *.pl*.

Decisioni di Progetto

La KB del sistema è costituita da diversi file dove ognuno rappresenta la conoscenza su diversi aspetti del dominio considerato. I file in questione sono i seguenti:

- housingFacts.pl, contiene i fatti circa gli alloggi a disposizione e le loro caratteristiche;
- calendar.pl, contiene tutti i fatti circa le disponibilità quotidiane degli alloggi rappresentati nel file precedente;
- transport.pl, contiene tutti i fatti sulle stazioni metropolitane e le loro caratteristiche;
- turismFacts.pl, contiene i fatti sulle attrazioni turistiche e le loro caratteristiche;
- rules.pl, contiene le varie regole che consentono di derivare ulteriore conoscenza.

Tutti fatti che vengono definiti sono stati definiti dai vari dataset utilizzati nel programma al fine di ottenere ad esempio la lista degli alloggi con i relativi servizi, le stazioni metropolitane e così via.

Come già anticipato, il tipo di rappresentazione adottato è quello basato su individui e relazioni.

Gli individui rappresentanti nella nostra KB sono i seguenti:

- housing(H), rappresenta un alloggio il cui identificativo è H;
- host(H), rappresenta un venditore il cui id è H;
- neighborhood(N), rappresenta un quartiere della città di nome N;
- attraction(A), rappresenta una determinata attrazione turistica della città, il cui identificativo è A;
- station(S), rappresenta una stazione metropolitana il cui identificativo è S;
- line(L), rappresenta una linea metropolitana identificata da L;
- date(G, M, A), rappresenta una determinata data di giorno G, mese M e anno A.

Per quanto riguarda le relazioni, abbiamo rappresentato sia le proprietà degli individui che le associazioni tra questi.

In generale, per la maggior parte delle proprietà si è utilizzato la rappresentazione del tipo individuo-proprietà-valore, mentre, per le proprietà booleane si è utilizzata la seguente notazione *proprietà(X)*, dove X è un individuo che gode della proprietà indicata. Questa differenza di notazione è stata adottata per distinguere quelle proprietà 'obbligatorie' di ciascun individuo da quelle opzionali (per esempio servizi fornito nell'alloggio).

<u>Proprietà alloggi</u>

- prop(X,'neighbourhood', N): indica che il quartiere dell'alloggio X è N;
- prop(X,'price', P): indica che il prezzo a notte dell'alloggio X è P;
- prop(X,'roomType', RT): indica che il tipo si stanza dell'alloggio X è RT;
- prop(X,'accommodates', A): indica che il numero di persone a cui è dedicato l'alloggio X è A;
- prop(X,'bath', B): indica che il numero di bagni presenti nell'alloggio X è B;
- prop(X,'bedrooms', B): indica che il numero di camera da letto nell'alloggio X è B;
- prop(X,'beds',B): indica che il numero di letti nell'alloggio X è B;
- prop(X,'host', H): indica che l'host dell'alloggio X è H;
- prop(X,'latitude', L): indica che la latitudine del luogo in cui si trova l'alloggio X è L;
- prop(X,'longitude', L): indica che la longitudine del luogo in cui si trova l'alloggio X è L;
- prop(X,'maxNights', M): indica che il numero massimo di notti consentite nell'alloggio X è M;
- prop(X,'minNights', M): indica che il numero minimo di notti da affittare nell'alloggio è M.
- prop(X, 'cluster', C): indica che l'alloggio X viene classificato nel cluster C;
- sharedBath(X): indica che l'alloggio X ha un bagno condiviso;
- hasWifi(X): indica che l'alloggio X ha il wifi;

- hasheating(X): indica che l'alloggio X ha il riscaldamento;
- hasAlarm(X): indica che l'alloggio X ha l'allarme;
- allowPets(X): indica che l'alloggio X consente l'ingresso agli animali;
- hasTV(X): indica che l'alloggio X ha almeno una TV;
- hasRefrigerator(X): indica che l'alloggio X possiede un frigorifero;
- hasElevator(X): indica che l'alloggio X ha l'ascensore;
- hasAirConditioning(X): indica che l'alloggio X possiede l'aria condizionata;
- hasParking(X): indica che l'alloggio X ha un parcheggio privato per i clienti.
- available(X, date(G, M, A)): indica che l'alloggio X è disponibile nella data indicata da date, di giorno G, mese M e anno A.

NB: per X si indica semplicemente l'ID dell'alloggio dato che sono unici e non si confondono con eventuali ulteriori identificativi presenti nel programma.

Proprietà quartieri

luxuryNeighborhood(N): indica che il quartiere N è un quartiere lussuoso;

turisticNeighborhood(N): indica che il quartiere N è un quartiere turistico.

NB: per N si indica il nome del quartiere.

Proprietà attrazioni turistiche

- prop(A, 'latitude', L): indica che l'attrazione turistica A si trova in latitudine L;
- prop(A, 'longitude', L): indica che l'attrazione turistica A si trova in longitudine L;
- prop(A, 'neighborhood', N): indica che l'attrazione turistica A si trova nel quartiere N.

NB: per A si indica semplicemente l'ID dell'attrazione in quanto distinguibile dagli altri identificativi.

Proprietà trasporto

- prop(S, 'latitude', Lat): indica che la stazione metropolitana S si trova nella latitudine Lat;
- prop(S, 'latitude', Lon): indica che la stazione metropolitana S si trova nella longitudine Lon;
- prop(S, 'name' N): indica che la stazione metropolitana S si chiama N;
- prop(L, 'name', N): indica che la linea metropolitana L si chiama N;
- connection(S1, S2, L) indica che le stazioni metropolitane S1 e S2 sono collegate tra loro tramite la linea L.

NB: per S, S1, S2 si indicano station(X) con X che rappresenta l'identificativo della stazione, mentre, con L si indica line(Y) con Y che rappresenta l'identificativo della linea metropolitana.

Tutti fatti definiti finora vengono interrogati tramite query per mostrare tale conoscenza all'utente e/o per eseguire determinate funzionalità.
Infatti, si interrogano i fatti circa gli alloggi nella funzionalità 'Ricerca Alloggio' dell'utente per cercare tutti gli alloggi che hanno le caratteristiche indicate dall'utente e per mostrare le proprietà degli alloggi risultanti dalla ricerca.
Sempre in questa funzionalità, l'utente ha la possibilità di vedere degli alloggi simili a quello corrente (grazie al clustering effettuato precedentemente). In questo caso, si effettua un apposita query per ottenere tutti gli alloggi appartenenti a quella classe.

```
results = kb.ask(f"prop('{current_number}', 'cluster', X).")
result_cluster = results[0]['X']
result_simili = kb.ask(f"prop(X,'cluster', '{result_cluster}').")
```

Mentre, i fatti sui trasporti vengono interrogati nel problema di ricerca per la costruzione del grafo di ricerca, che corrisponde alla rete ferroviaria.

```
stations = kb.ask('station(X).')
nodes = []
arcs = []
for item in stations:
      nodes.append(item['X'])
results = kb.ask('connection(Station1, Station2, _).')
```

Infatti, in questo modo otteniamo le informazioni su tutte le stazioni presenti, ma anche i collegamenti che ci sono tra queste.
Si effettuano delle ulteriori query per poter avere una stampa del percorso trovato più esplicativa.

Fatti sulle date

daysPerMonth(M, N): indica che nel mese M il numero di giorni presenti è N.

Clausole definite:

La base di conoscenza presente diverse regole che permettono di derivare ulteriore conoscenza sul dominio.

Sono state definite le seguenti regole:

- *isInTuristNeigh(H, N):- housing(H), turisticNeighbourhood(N), prop(H, 'neighbourhood', N):* verifica se l'alloggio H si trova in un quartiere turistico;
- *isInLuxuryNeigh(H, N):- housing(H), luxuryNeighbourhood(N), prop(H, 'neighbourhood', N):* verifica che l'alloggio H si trova in quartiere
- *isNearAttraction(H, T):- housing(H), attraction(T), distance(H, T, D), D < 2000*: indica se l'alloggio è a meno di 2 chilometri di distanza da un'attrazione turistica;
- *sameNeigh(X1, X2):- prop(X1, 'neighbourhood', N), prop(X2, 'neighbourhood', N):* verifica che due elementi (alloggi/attrazioni) si trovano nello stesso quartiere
- *distance(H, T, D):- prop(H, 'latitude', Lat1), prop(H, 'longitude', Lon1),prop(T, 'latitude', Lat2), prop(T, 'longitude', Lon2), calculate_distance(Lat1, Lon1, Lat2, Lon2, Distance), D is Distance:* regola che calcola la distanza tra due posizioni (alloggio/attrazione/stazione);
- *calculate_distance(Lat1, Lon1, Lat2, Lon2, Distance) :- DeltaLat is abs(Lat1 - Lat2), DeltaLon is abs(Lon1 - Lon2), Distance is sqrt(DeltaLat**2 + DeltaLon**2) * 111319.9 /1000*: regola che consente il calcolo della distanza in chilometri tra due punti in linea d'aria;
- *date(D, M, Y):- Y >= 2023, M >=1, M =< 12, daysPerMonth(M, X), D >= 1, D =< X*: regola che verifica la validità di una data;
- *nextDay(date(D, M, Y), date(ND, NM, NY)) :- daysPerMonth(M, X), D < X , ND is D + 1, NM = M, NY = Y;*
- *nextDay(date(D, M, Y), date(1, NM, NY)):- daysPerMonth(M, X), D = X, NM is M + 1, NY = Y;*
- *nextDay(date(31, 12, Y), date(1, 1, NY)):- NY is Y + 1:* queste tre regole nextDay servono per calcolare la data successiva e coprono i vari casi che si possono verificare;
- *journey(ID, StartDate, Duration):- checkAvailability(ID, StartDate, Duration)*: regola che verifica se è possibile alloggiare in un certo alloggio per un certo periodo a partire da una determinata data in base alle disponibilità giornaliere dell'alloggio;
- *checkAvailability(_, _, 0).*
- *checkAvailability(ID, Date, Duration):- available(ID, Date), nextDay(Date, NextDate), NewDuration is Duration - 1, checkAvailability(ID, NextDate, NewDuration):* le regole checkAvailability vanno a verificare in maniera ricorsiva le disponibilità giornaliere di un certo alloggio.

Le regole sopra indicate vengono utilizzate per vari scopi nel programma, come aggiungere informazioni nel dataset, calcolare l'euristica nel problema di ricerca, verificare la disponibilità di un certo alloggio in un determinato periodo, …

APPRENDIMENTO SUPERVISIONATO

Sommario

In generale, l'apprendimento supervisionato prevede che ci sia un set di esempi descritti in termini di feature di input e feature target (di solito è solo una). L'obiettivo è quello di prevedere la feature target per nuovi esempi. A tale scopo, il dataset viene suddiviso in Training Set e Test Set. Sul Training Set si addestra un modello di predizione, il quale verrà valutato e testato sul Test Set. Infatti, nel Training Set è conosciuto il valore della feature target, mentre, nel Test Set questo è sconosciuto [1]. Ci sono diversi tipi di modelli predittivi che possiamo usare a tale scopo e nel nostro sistema si ricerca il modello e la relativa configurazione che comporta un miglior apprendimento e di conseguenza una maggiore accuratezza nelle predizioni. Nel nostro caso, l'obiettivo è quello di prevedere la classe di prezzo a cui un certo alloggio appartiene (Economic, Meddium, Premium) dove ogni classe corrisponde a una range differente di prezzo a notte.

Strumenti utilizzati

Per il raggiungimento del nostro obiettivo, sono stati utilizzati diversi moduli forniti dalla libreria sklearn. In particolare, sono stati utilizzati *train_test_split* che consente la suddivisione del dataset tra Training Set e Test Set, GridSearchCV che ha consentito la ricerca del modello migliore, StandardScaler che effettua lo scaling del dataset e inoltre i vari moduli corrispondenti ai modelli provati (RandomForestClassifier, DecisionTreeClassifier, GradientBoostingClassifier, MLPClassifier, GaussianNB) e metriche utilizzate per la loro valutazione (Precision, Recall, F1).

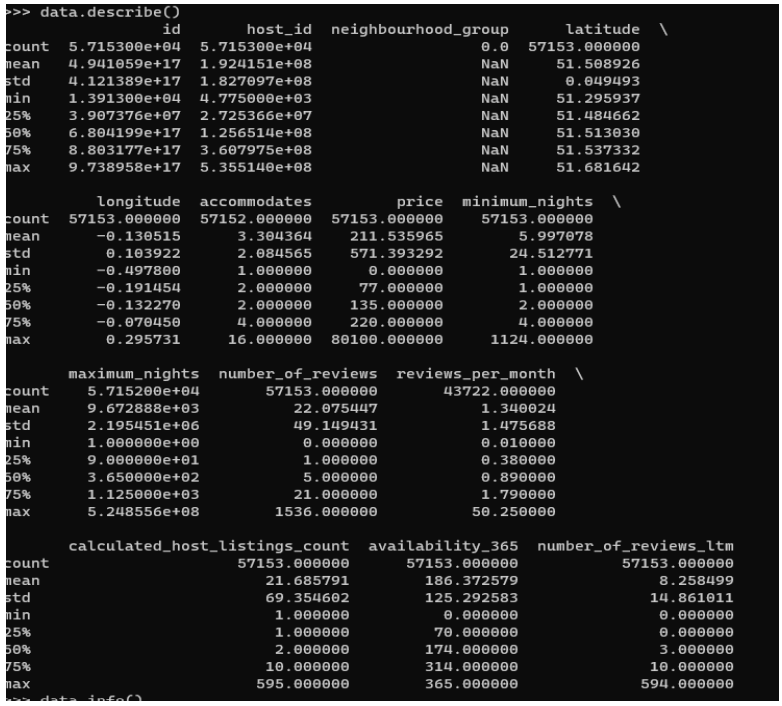
Decisioni di Progetto

Innanzitutto, è stato effettuato il Data Cleaning, ovvero la pulizia del dataset da eventuali errori presenti. Il dataset utilizzato per l'apprendimento supervisionato è datasetListings.csv.

Data Cleaning

Sono state effettuate delle analisi preliminari per conoscere il tipo di dati contenuto nel dataset e conoscere quali modifiche bisogna effettuare.

#	Column
0	id
1	name
2	host_id
3	host_name
4	host_since
5	host_response_rate
6	host_response_time
7	host_is_superhost
8	neighbourhood
9	neighbourhood_group
10	latitude
11	longitude
12	room_type
13	accommodates
14	amenities
15	price
16	minimum_nights
17	maximum_nights
18	number_of_reviews
19	last_review
20	reviews_per_month
21	calculated_host_listings_count
22	availability_365
23	number_of_reviews_ltm
24	license



```
>>> data.describe()
                 id       host_id  neighbourhood_group      latitude  \
count  5.715300e+04  5.715300e+04                  0.0  57153.000000
mean   4.941059e+17  1.924151e+08                  NaN     51.508926
std    4.121389e+17  1.827097e+08                  NaN      0.049493
min    1.391300e+04  4.775000e+03                  NaN     51.295937
25%    3.907376e+07  2.725366e+07                  NaN     51.484662
50%    6.804199e+17  1.256514e+08                  NaN     51.513030
75%    8.803177e+17  3.607975e+08                  NaN     51.537332
max    9.738958e+17  5.355140e+08                  NaN     51.681642

          longitude  accommodates         price  minimum_nights  \
count  57153.000000  57152.000000  57153.000000    57153.000000
mean      -0.130515      3.304364    211.535965        5.997078
std        0.103922      2.084565    571.393292       24.512771
min       -0.497800      1.000000      0.000000        1.000000
25%       -0.191454      2.000000     77.000000        1.000000
50%       -0.132270      2.000000    135.000000        2.000000
75%       -0.070450      4.000000    220.000000        4.000000
max        0.295731     16.000000  80100.000000     1124.000000

       maximum_nights  number_of_reviews  reviews_per_month  \
count    5.715200e+04       57153.000000       43722.000000
mean     9.672888e+03          22.075447           1.340024
std      2.195451e+06          49.149431           1.475688
min      1.000000e+00           0.000000           0.010000
25%      9.000000e+01           1.000000           0.380000
50%      3.650000e+02           5.000000           0.890000
75%      1.125000e+03          21.000000           1.790000
max      5.248556e+08        1536.000000          50.250000

       calculated_host_listings_count  availability_365  number_of_reviews_ltm
count                    57153.000000      57153.000000           57153.000000
mean                        21.685791        186.372579               8.258499
std                         69.354602        125.292583              14.861011
min                          1.000000          0.000000               0.000000
25%                          1.000000         70.000000               0.000000
50%                          2.000000        174.000000               3.000000
75%                         10.000000        314.000000              10.000000
max                        595.000000        365.000000             594.000000
>>> data.info()
```

Sono stati eliminati delle colonne inutili o contenente dei valori errati per l'obiettivo da raggiungere come NeighborhoodGroup, ReviewPerMonth, HostResponseRate e License, Id, HostId, HostName, HostResponseRate, HostIsSuperhost, Latitude, Longitude, Name e Amenities. Più precisamente, quest'ultime due colonne, prima di essere rimosse, sono state utilizzate per ottenere ulteriori informazioni che sono state aggiunte al dataset tramite nuove colonne.

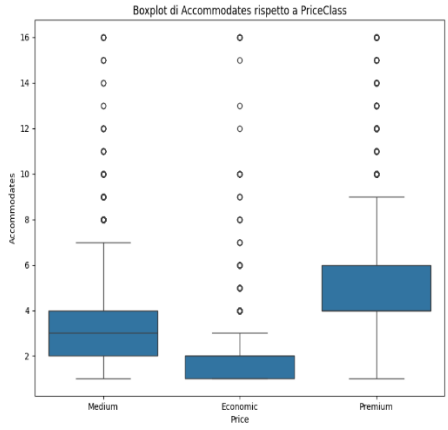
Inoltre, sono state aggiunte anche due nuove colonne grazie a query effettuate sulla KB, ovvero inTuristicNeighborhood e inLuxuryNeighborhood che indicano se un alloggio si trova in un quartiere turistico e lussuoso rispettivamente.

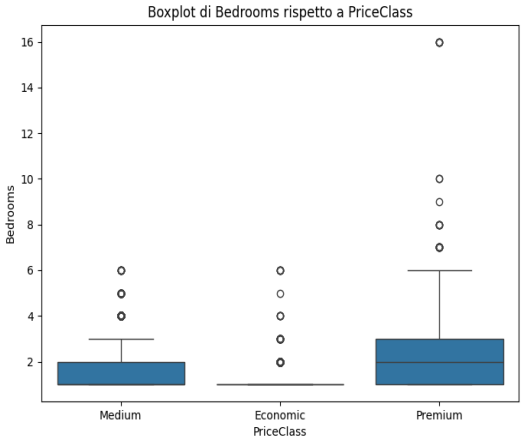
(È stata eseguita, in realtà, anche le query per capire se l'alloggio è vicino a una qualsiasi attrazione turistica, ma il numero di alloggi a cui dava un riscontro positivo era di cica 40, per cui si è deciso di non aggiungere tale informazione.)

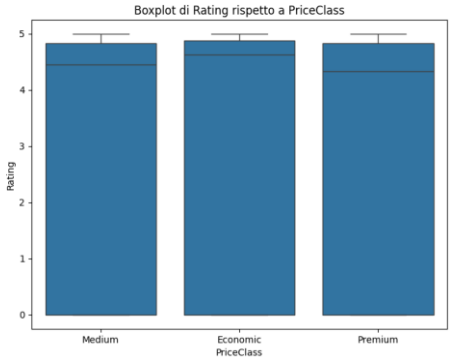
Anche la colonna Price è stata eliminata, ma viene prima utilizzata per indicare le varie classi di prezzo definendo la colonna PriceClass, che può contenere:

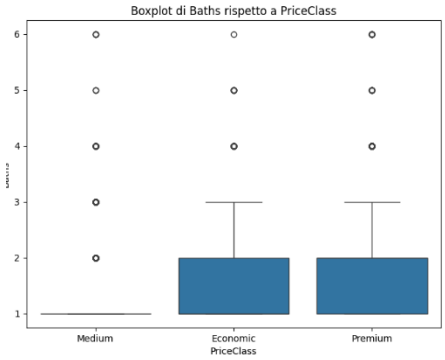
- Economic, prezzi fino a 75;
- Medium, prezzi tra 75 e 205;
- Premium, prezzi superiori a 205.

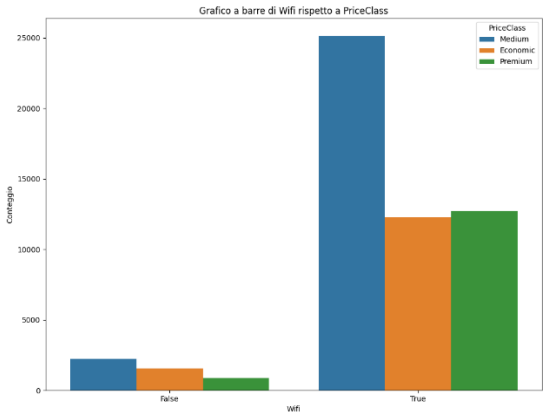
Sono state effettuate delle ulteriori analisi per cercare di capire quanto le restanti variabili possono influenzare la classe di prezzo.

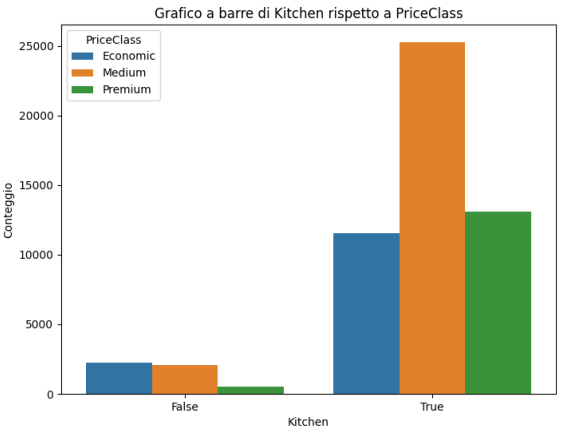












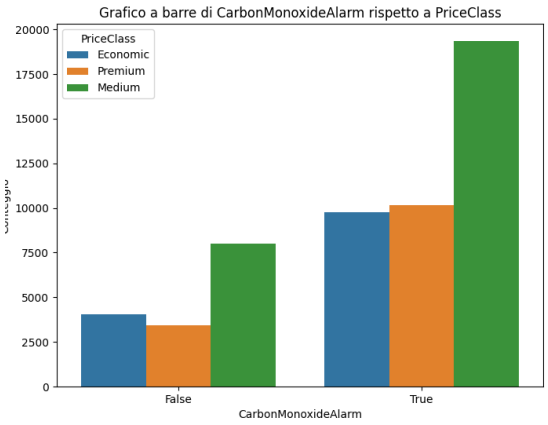


Grafico a barre di CarbonMonoxideAlarm rispetto a PriceClass
PriceClass
Economic
Premium
Medium
Conteggio
CarbonMonoxideAlarm
False
True

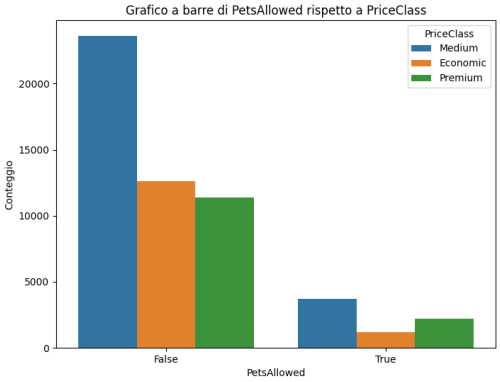


Grafico a barre di PetsAllowed rispetto a PriceClass
PriceClass
Medium
Economic
Premium
Conteggio
PetsAllowed
False
True

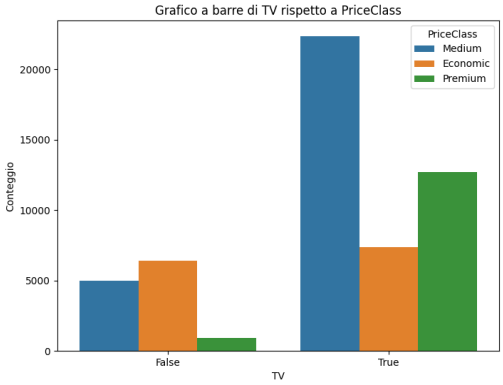


Grafico a barre di TV rispetto a PriceClass
PriceClass
Medium
Economic
Premium
Conteggio
TV
False
True

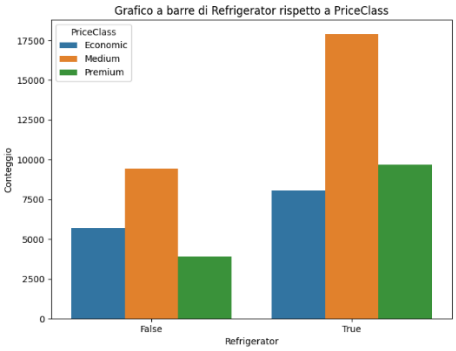


Grafico a barre di Refrigerator rispetto a PriceClass
PriceClass
Economic
Medium
Premium
Conteggio
Refrigerator
False
True

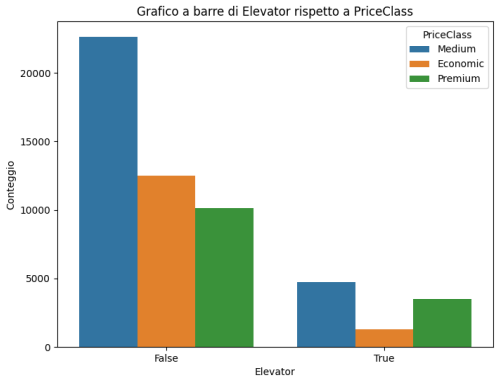


Grafico a barre di Elevator rispetto a PriceClass
PriceClass
Medium
Economic
Premium
Conteggio
Elevator
False
True

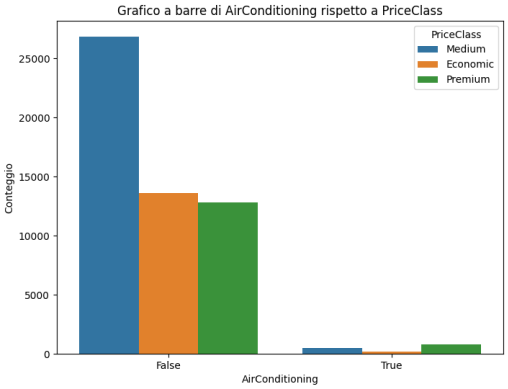


Grafico a barre di AirConditioning rispetto a PriceClass
PriceClass
Medium
Economic
Premium
Conteggio
AirConditioning
False
True

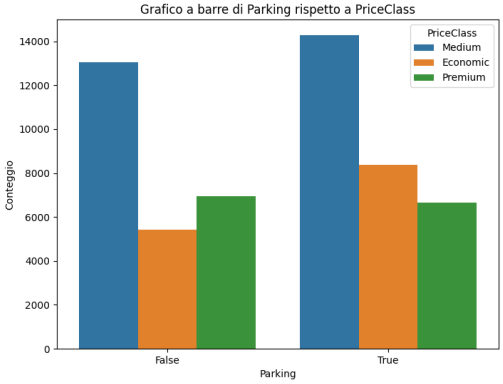


Grafico a barre di Parking rispetto a PriceClass
PriceClass
Medium
Economic
Premium
Conteggio
Parking
False
True

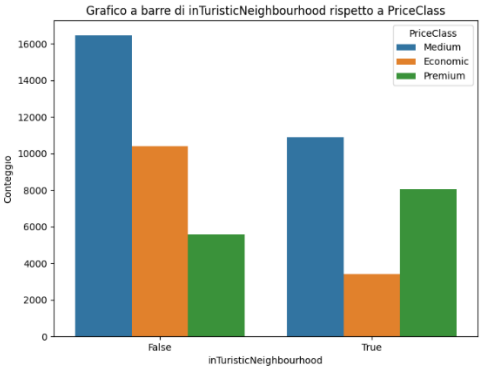


Grafico a barre di inTuristicNeighbourhood rispetto a PriceClass
PriceClass
Medium
Economic
Premium
Conteggio
inTuristicNeighbourhood
False
True

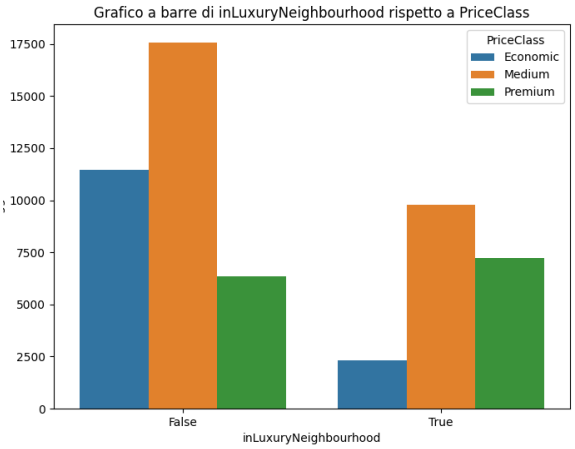


Grafico a barre di inLuxuryNeighbourhood rispetto a PriceClass
PriceClass
Economic
Medium
Premium
17500
15000
12500
10000
7500
5000
2500
0
False
True
inLuxuryNeighbourhood

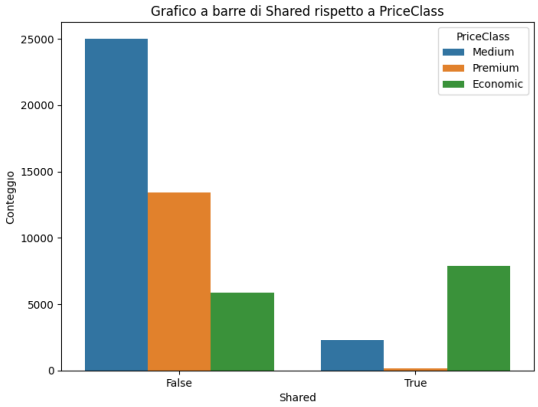
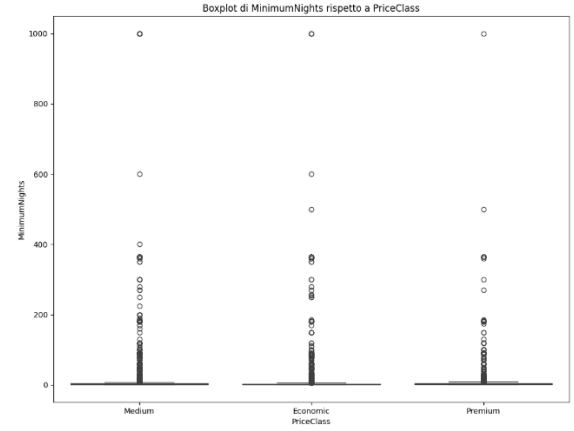
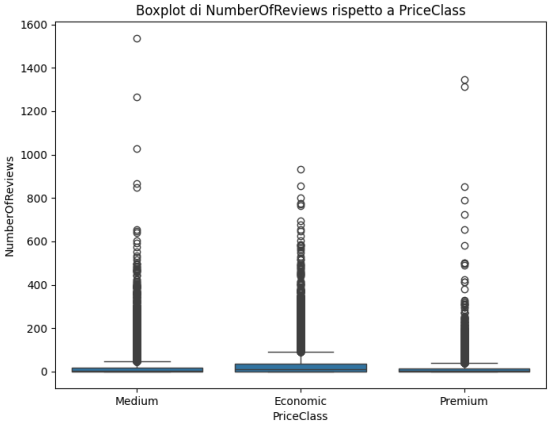
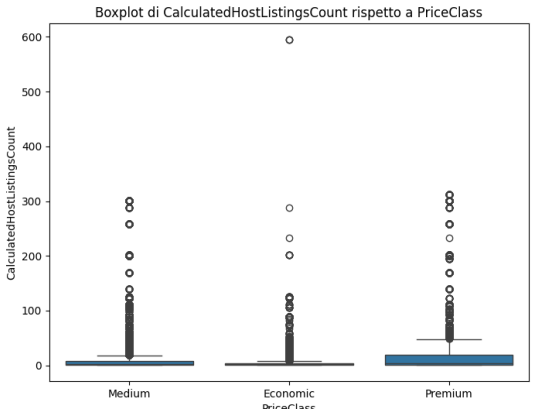
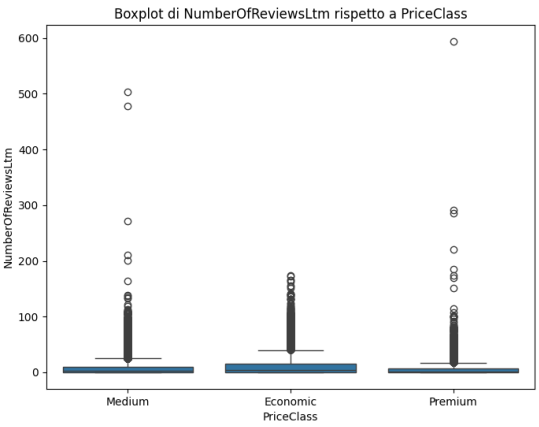


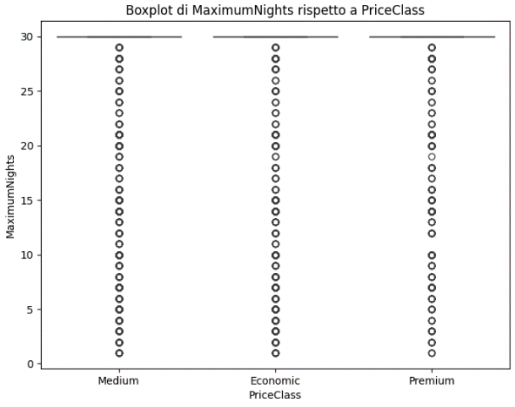
Grafico a barre di Shared rispetto a PriceClass
PriceClass
Medium
Premium
Economic
25000
20000
15000
10000
5000
0
Conteggio
False
True
Shared


Boxplot di MinimumNights rispetto a PriceClass
1000
800
600
400
200
0
MinimumNights
Medium
Economic
Premium
PriceClass


Boxplot di NumberOfReviews rispetto a PriceClass
1600
1400
1200
1000
800
600
400
200
0
NumberOfReviews
Medium
Economic
Premium
PriceClass


Boxplot di CalculatedHostListingsCount rispetto a PriceClass
600
500
400
300
200
100
0
CalculatedHostListingsCount
Medium
Economic
Premium


Boxplot di NumberOfReviewsLtm rispetto a PriceClass
600
500
400
300
200
100
0
NumberOfReviewsLtm
Medium
Economic
Premium
PriceClass


Boxplot di MaximumNights rispetto a PriceClass
30
25
20
15
10
5
0
MaximumNights
Medium
Economic
Premium
PriceClass

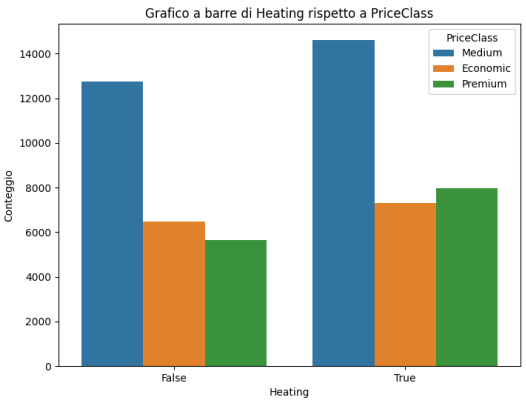
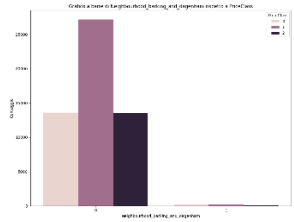
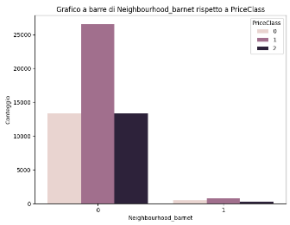
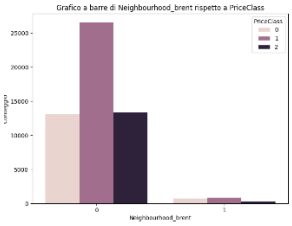


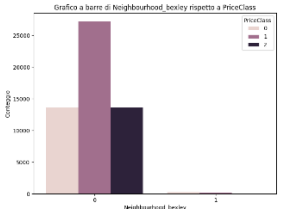
Grafico a barre di Heating rispetto a PriceClass
PriceClass
Medium
Economic
Premium
14000
12000
10000
8000
6000
4000
2000
0
Conteggio
False
True
Heating

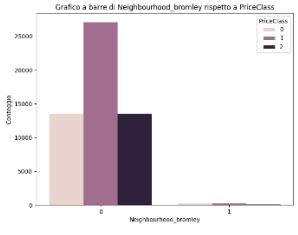
Analisi influenza quartieri e room type:

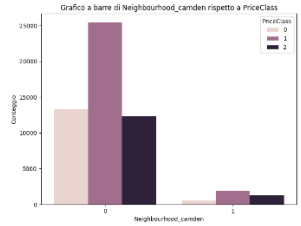


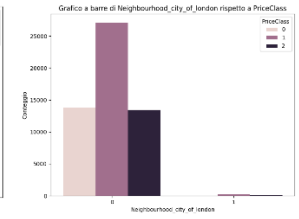


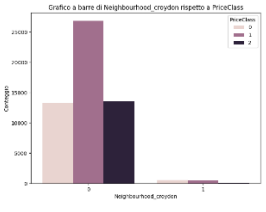


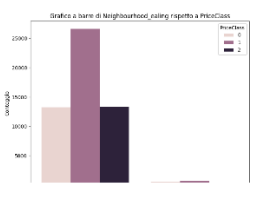


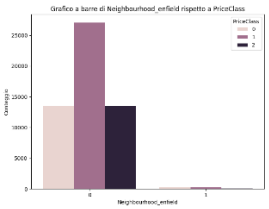


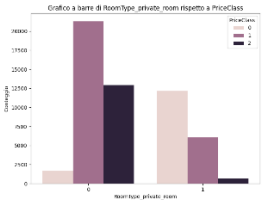


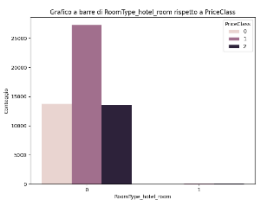


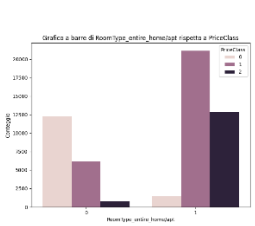


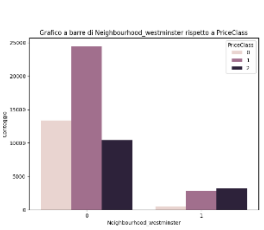


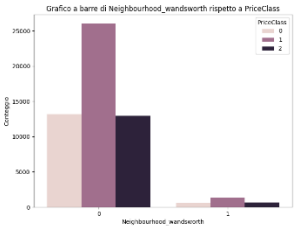


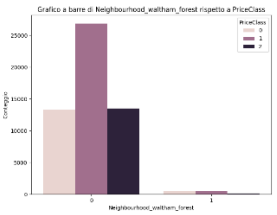


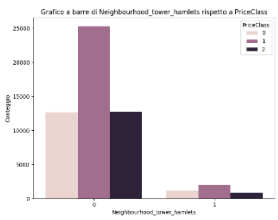


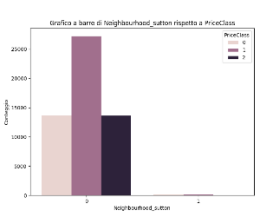


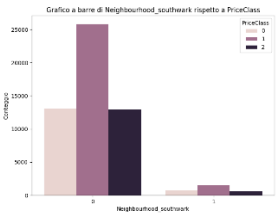


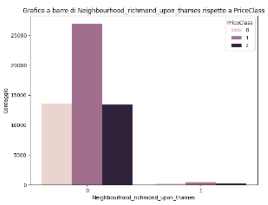


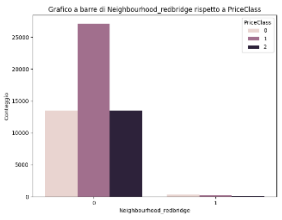


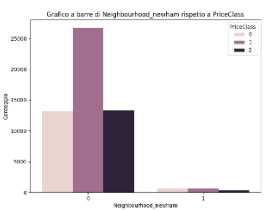


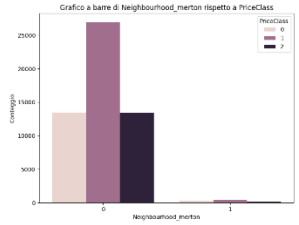


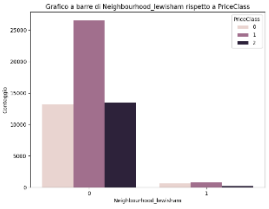


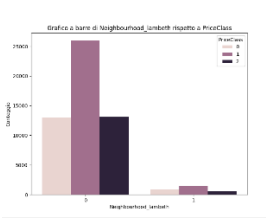


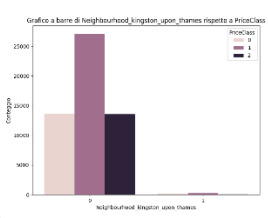


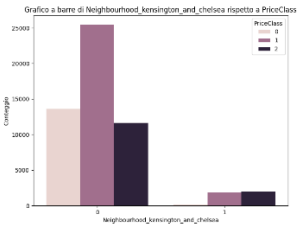


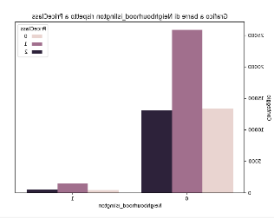


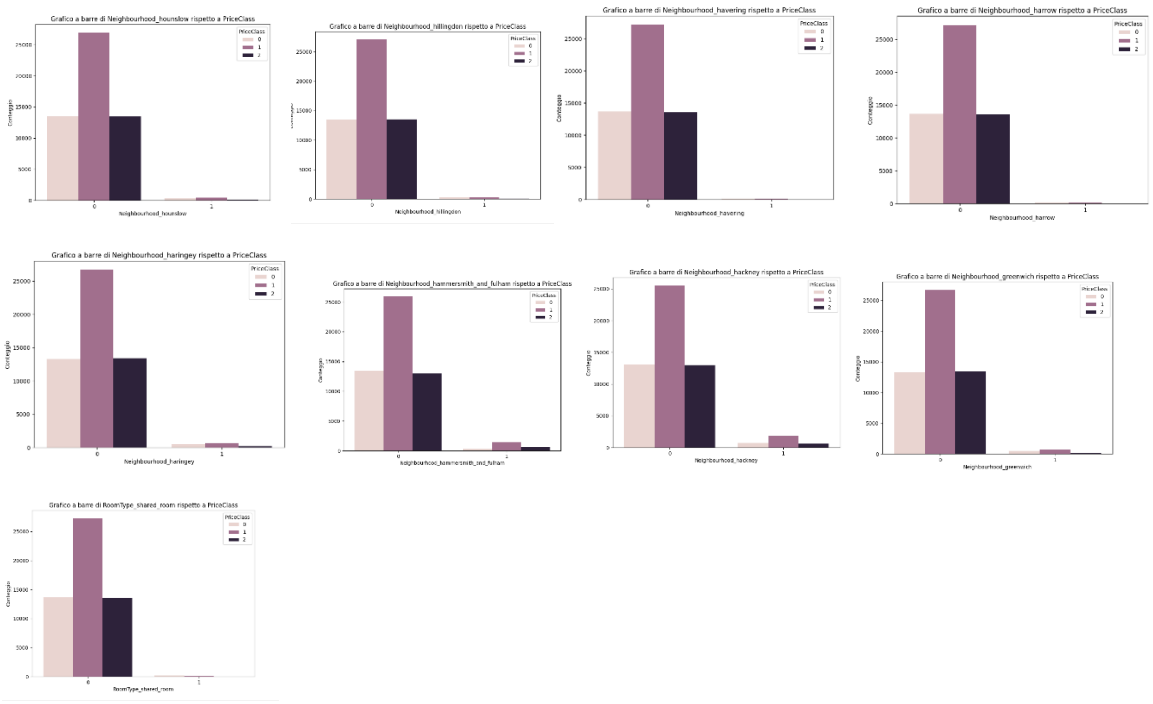












In seguito a questa analisi, abbiamo deciso di rimuovere le variabili Availability365, MinimumNights, MaximumNights, NumberOfReviews, CalculatedHostListingsCount e NumberOfReviewsLtm, Beds, Rating, poiché si è notato che non sono in grado di distinguere la classe di prezzo per cui avrebbero una bassa influenza.

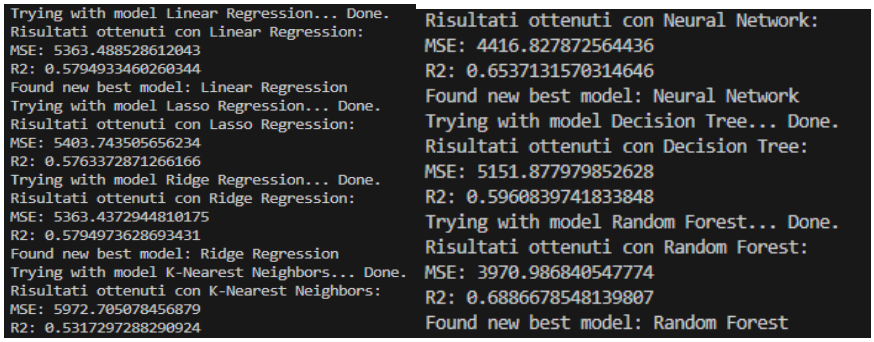
Inoltre, i dati numerici sono stati scalati mediante StandardScaler, mentre, le variabili discrete, ovvero RoomType e Neighborhood) sono state elaborate mediante l'approccio del One-Hot Encoding che consiste nel sostituire la variabile in tante variabile binarie quanti sono i valori del dominio e assumerà valore vero solo una di queste variabili, ovvero la variabile corrispondente al valore effettivo.

Infine, dell'intero dataset l'80% è stato destinato per la fase di Training, mentre, il restante 20% è stato destinato per la fase di Testing.

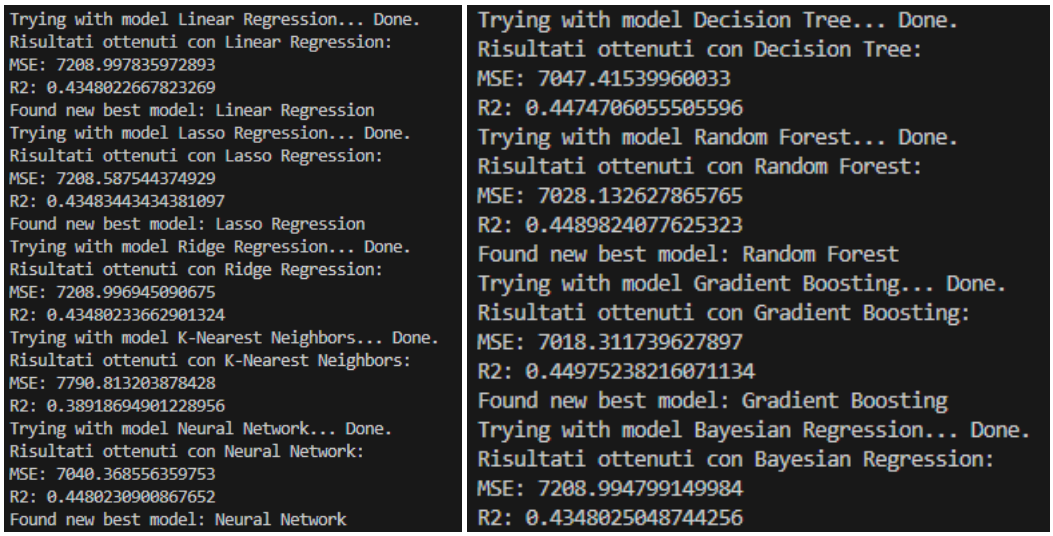
Valutazione

Si è provato a definire inizialmente un modello di previsione del prezzo esatto degli alloggi, ma nonostante siano stati provati diversi modelli ed effettuati diversi esperimenti l'errore quadratico dei modelli era molto alto. I risultati ottenuti sono infatti i seguenti:

<u>Tutte le variabili:</u>



<u>Con variabili relative a servizi e quartieri rimosse:</u>



Nella previsione del prezzo esatto, i modelli utilizzati sono stati:

Linear Regression, Lasso Regression, Ridge Regression, K-NN, Neural Network, Decision Tree, Random Forest, Gradient Boosting e Bayesian Regression.

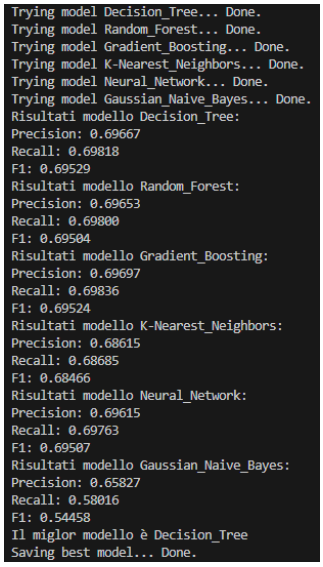
Questo errore elevato e scarsa capacità di adattamento del modello ai dati può essere dovuto da un enorme variazione dei prezzi degli alloggi tra loro, per cui non vi è una vera e propria regola che li definisce.

Invece, per la previsione l'apprendimento funziona nettamente meglio. L'obiettivo è prevedere la classe di prezzo, ovvero la fascia di prezzo, a cui appartiene un alloggio in base alle sue caratteristiche. Trattandosi, infatti, di un intervallo di valori e non di valori esatti, è stato possibile addestrare un modello piuttosto accurato.

Le metriche di valutazione utilizzate sono Precision, Recall e F1. La scelta del modello migliore viene effettuata proprio in base al valore di F1, ovvero la media armonica tra precision e recall. Come opzione di valutazione usiamo weighted, poiché la distribuzione delle classi è sbilanciata; quindi, in questo modo andiamo a dare maggiore peso alle classi più popolate,

per cui siamo in grado di avere una comprensione complessiva delle prestazioni di un modello in modo equo rispetto alle dimensioni delle classi.

Nella ricerca del modello migliore sono stati provati i seguenti modelli:



- **DecisionTree**:
  In cui si ricerca il miglior parametro:
  maxDepth tra i valori 10, 20 e 30.

Precision	0.69667
Recall	0.69818
F1	0.69529

- **RandomForest**:
  In cui si cerca la configurazione migliore del parametro:
  n_estimator tra i valori 10, 50 e 100

Precision	0.69653
Recall	0.69800
F1	0.69504

- **GradientBoosting**:
  In cui si cerca la configurazione migliore del parametro:
  n_esimator tra i valori 10, 50 e 100

Precision	0.69697
Recall	0.69836
F1	0.69524

- **K-NearestNeighbors**:

In cui si cerca la configurazione migliore dei parametri:
n_neighbors tra i valori 3, 5 e 7

Precision	0.68615
Recall	0.68685
F1	0.68466

- **NeuralNetwork**:
  In cui si cerca la configurazione migliore dei parametri:
  hidden_layer_size tra I valori (50,), (100, 50), (100, 100)
  activation tra I valori relu e tahn
  alpha tra i valori 0.0001, 0.001 e 0.01

Precision	0.69615
Recall	0.69763
F1	0.69507

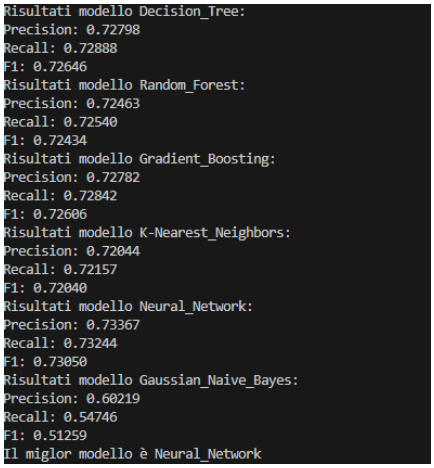
- **GaussianNaiveBayes**:

Precision	0.65827
Recall	0.58016
F1	0.54458

Per questo tipo di esperimento, sono state eliminate le variabili relative ai servizi e ai quartieri. I vari modelli funzionano mediamente bene e il modello migliore è Decision Tree, mentre, quello peggiore è il classificatore Naive Bayes.

L'esperimento successivo è stato effettuato considerando anche le variabili relative ai vari servizi e i quartieri e abbiamo ottenuto in questo modo un maggior valore di F1 nei vari modelli.

Sono stati provati gli stessi modelli con le stesse configurazioni dell'esperimento precedente:



- **DecisionTree**:

Precision	0.72798
Recall	0.72888
F1	0.72646

- **RandomForest**:

Precision	0.72463
Recall	0.72540
F1	0.72434

- **GradientBoosting**:

Precision	0.72782
Recall	0.72842
F1	0.72606

- **K-NearestNeighbors**:

Precision	0.72044
Recall	0.72157
F1	0.72040

- **NeuralNetwork**:

Precision	0.73367
Recall	0.73244
F1	0.73050

- **GaussianNaiveBayes**:

Precision	0.60219
Recall	0.54746
F1	0.51259

Complessivamente, possiamo concludere che i modelli (tranne il Naive Bayes) funzionano meglio con un valore di F1 maggiore del 0.7 e quindi l'attività di classificazione viene effettuata correttamente per la maggior parte degli esempi. Il modello migliore in questo caso la rete neurale con un valore di F1 pari a 0.73050.

NB: Infine, il modello migliore ottenuto in questa fase, viene salvato per poter essere utilizzato nella predizione della classe di prezzo di esempi che vengono forniti manualmente dall'utente per la funzionalità 'Predizione classe di prezzo'.

Apprendimento non supervisionato

Sommario

Nell'apprendimento non supervisionato, rispetto a quello supervisionato, le feature target non sono fornite dagli esempi. L'obiettivo è di costruire una classificazione naturale per i dati. Un metodo generale per questo tipo di apprendimento è il clustering, il quale è stato utilizzato per questo progetto. In generale, il clustering può essere di due tipi: hard clustering e soft clustering [2]. Nel nostro caso è stata adottata la prima tipologia con conseguente utilizzo del K-means per partizionare i vari alloggi in classi. L'obiettivo infatti è quello di raggruppare tra loro gli alloggi che sono più simili, in modo tale che si può mostrare all'utente gli alloggi che sono simili a uno piaciuto. Quindi l'idea sarebbe quella di realizzare una sorta di Recommendation System.

Strumenti utilizzati

Per lo svolgimento di tale task sono stati utilizzati i moduli learnKMeans e learnProblem forniti da AiPython del libro Artificial Intelligence. Sono state effettuate alcune modifiche per

adattarli alle nostre esigenze. Mentre, per rappresentare graficamente il grafico è stata utilizzata la libreria matplotlib.

Decisioni di Progetto

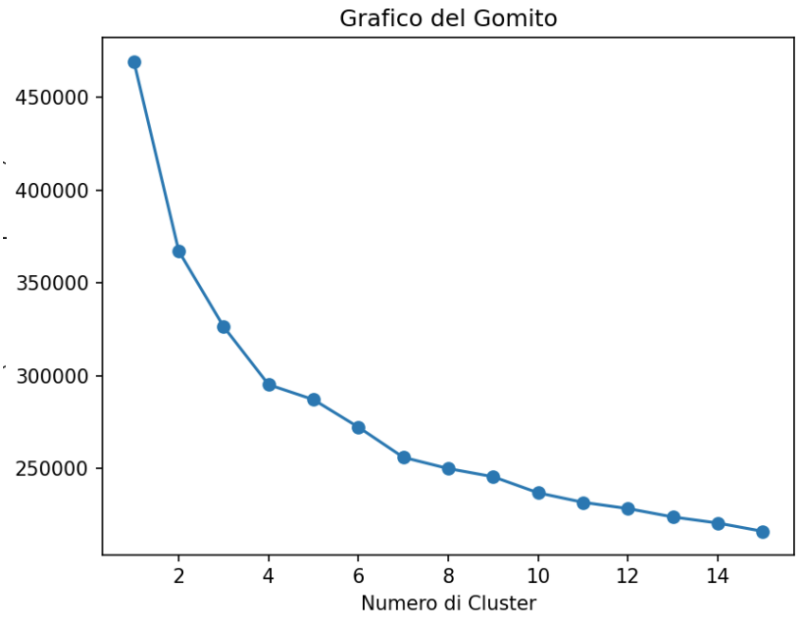
La tecnica del clustering è stata utilizzata per poter individuare e raggruppare alloggi che presentano similarità tra loro, salvando i relativi risultati come fatti nella KB.

Questi verranno utilizzati per poter mostrare gli alloggi simili, ovvero gli alloggi che appartengono alla stessa classe. Infatti, gli utenti nella funzionalità 'Ricerca alloggi' possono cercare gli alloggi simili a uno che è particolarmente piaciuto.

Per ottenere il numero ideale di cluster è stata applicata la regola del gomito. A tal proposito, è stato modificato il modulo fornito per consentire l'applicazione di tale regola. Infatti, nel metodo elbow_rule si va a effettuare il clustering per 15 volte incrementando il numero di cluster per ogni iterazione. Una volta terminato, viene mostrato il grafico e l'utente sceglie il numero di cluster che, secondo lui, corrisponde al gomito della curva. Dopodiché, si effettua il clustering con il valore scelto e i risultati vengono salvati nella KB.

Un'ultima considerazione riguarda le variabili utilizzate per effettuare il clustering. Bisogna osservare che sono state effettuate le stesse modifiche ed eliminazioni di variabili viste nell'apprendimento supervisionato. Tuttavia, in questo caso sono state mantenute le variabili Baths e Rating ed è stato aggiunto anche PriceClass.

Valutazione



Questo grafico rappresenta l'esperimento che abbiamo effettuato circa il clustering applicando la regola del gomito. Dalla curva ottenuta è piuttosto difficile individuare un gomito esplicito, ma possiamo considerare il valore 7 come numero ideale di cluster. Infatti, oltre tale valore la curva tende ad appiattirsi e la riduzione dell'errore non è significativa.

Problema di ricerca

Sommario

Il problema di un agente che decide cosa fare può essere considerato come il problema della ricerca di un percorso in un grafo. L'idea di ricerca è semplice: l'agente costruisce un insieme di soluzioni parziali a un problema che vengono verificate per veder se sono veramente soluzioni o se potrebbero portare a soluzioni. Quindi, dato un arco uscente da un nodo, verifica se questo costituisce un percorso verso uno degli obiettivi. Se sì stop, altrimenti costruisce nuove soluzioni parziali estendendo quella corrente nei diversi nodi raggiungibili [5]. Nel nostro caso, l'obiettivo è quello di cercare il percorso meno costoso tra i vari percorsi che rappresentano la rete metropolitana.

Strumenti utilizzati

Sono stati utilizzati i moduli searchGeneric, searchMPP, searchProblem e display, ovvero, moduli forniti da Aipython del libro Artificial Intelligence e sono state effettuate alcune modifiche per adattarle alle nostre esigenze.

Decisioni di Progetto

L'algoritmo di ricerca che viene eseguito è l'algoritmo di ricerca A*, il quale garantisce di trovare la soluzione ottimale mediante l'utilizzo dell'euristica e della funzione di costo del percorso. L'euristica considerata consiste nella distanza tra il nodo corrente (stazione corrente) fino al nodo obiettivo (stazione di destinazione).
Poiché il grafo corrisponde alla rete metropolitana e quindi ci sono percorsi bidirezionali, il grafo presenta dei cicli, ma anche dei percorsi multipli che possono causare un rallentamento della ricerca della soluzione. Per tale motivo, alla ricerca A* è stata integrata la tecnica del

MPP che va a scartare si ai percorsi multipli che i cicli nel grafo semplificando lo spazio di ricerca.

Inoltre, le modifiche effettuate rispetto ai moduli originari riguardano in particolar modo la stampa del percorso soluzione. Infatti, si è disposto un metodo che salva la soluzione migliore trovata e poi nel metodo di stampa si è fatto in modo che i nodi coinvolti nel percorso, nodo di partenza e nodo di destinazione vengono colorati diversamente in modo da distinguerli visivamente rispetto agli altri.

La ricerca viene utilizzata nella funzionalità 'Ricerca percorso' per permettere all'utente di cercare il percorso da fare, ma anche nella funzionalità 'Ricerca alloggi' per consentire di capire che percorso fare per raggiungere un certo alloggio partendo una stazione fittizia.

Valutazione

Si è iniziato col provare A* senza la tecnica del MPP, ma poiché il grafo presenta molti percorsi e anche dei cicli si è stato in grado di testare la ricerca di un percorso di dimensione 2.

A*

```
 1237 paths have been expanded and 2454 paths remain in the frontier
Solution path:  42 --> 183
Tempo di ricerca: 0.25670504570007324 secondi.
```

Incrementando la dimensione della soluzione a 6, l'algoritmo impiegava troppo tempo per trovare la soluzione ideale (più di 10 minuti), per cui non sono stati effettuati più degli esperimenti senza la tecnica del MPP.

Gli esperimenti successivi sono stati fatti sempre integrando nella ricerca A* la tecnica del MPP. Abbiamo fatto degli esperimenti confrontando le prestazioni dell'algoritmo quando utilizza l'euristica e quando invece non la utilizza. I risultati sono i seguenti:

CONFRONTO RICERCA EURISTICA

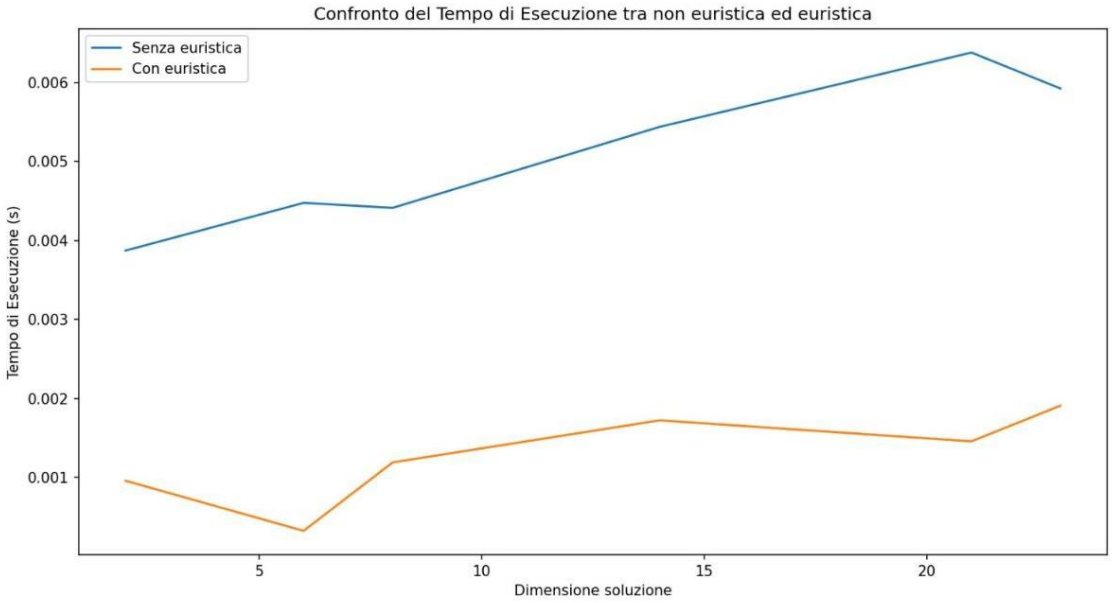
Dimensione	Ricerca senza euristica	Ricerca con euristica
2	0.000581	0.000303
6	0.000354	0.000448
8	0.004171	0.000429
14	0.004799	0.001048
21	0.006415	0.001046

23	0.006081	0.001376

Come si può vedere dalla tabella le prestazioni dell'A* con l'MPP sono nettamente migliori rispetto al caso precedente.

Notiamo anche che la ricerca mediante l'utilizzo dell'euristica risulta essere migliore rispetto al caso in cui non si utilizza e questa superiorità resta tale anche all'aumentare della dimensione della soluzione.

Analisi del grafico



L'algoritmo senza euristica ha una performance inferiore rispetto a quella con l'euristica qualunque sia la dimensione della soluzione.

Inoltre, possiamo notare anche che la curva dell'algoritmo senza euristica tende ad aumentare (le prestazioni peggiorano), mentre, quella con euristica non cresce nella stessa misura. Ovviamente, questa differenza di prestazioni si può notare solo con dati e grafici, ma non si nota durante l'esecuzione dato che si parla di millisecondi.

Conclusione: l'utilizzo dell'euristica integrata alla tecnica del MPP risulta essere la scelta ideale. Pertanto, nelle funzionalità di ricerca fornite all'utente si applica proprio la ricerca A* con tecnica del MPP e utilizzando l'euristica specificata sopra.

Conclusioni

Possiamo concludere che questo sistema potrebbe rappresentare la base per lo sviluppo di un sistema che assiste un viaggiatore in tutti gli aspetti di un viaggio. In questo modo, potrebbe evitare di utilizzare più applicazioni, ma avere tutto nella mani di una sola. Inoltre, l'utente potrebbe anche farsi un'idea del budget necessario per effettuare un certo tipo di viaggio e in un determinato luogo.

Questo programma si potrebbe estendere considerando ulteriori città oltre Londra e potrebbe includere ulteriori funzionalità che si aggiungono e/o estendono quelle esistenti. Per esempio, si potrebbe consentire direttamente la prenotazione degli alloggi aggiornando di conseguenza la KB, si potrebbe aggiungere ulteriori mezzi di trasporto e integrare anche lo spostamento a piedi.

Riferimenti Bibliografici

[1] https://artint.info/2e/html2e/ArtInt2e.Ch7.html

[2] https://artint.info/2e/html2e/ArtInt2e.Ch10.html

[3] https://artint.info/2e/html2e/ArtInt2e.Ch5.html

[4] https://artint.info/2e/html2e/ArtInt2e.Ch13.html

[5] https://artint.info/2e/html2e/ArtInt2e.Ch3.html