UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO | DIPARTIMENTO DI INFORMATICA

HOTEL BOOKING PREDITION

Documentazione Caso di Studio

AA 2022/2023

STUDENTESSA: Adriana Frascella, 735263, a.frascella11@studenti.uniba.it

REPOSITORY GitHub: https://github.com/adrianafrascella/ICON.git

DATASET: https://www.kaggle.com/datasets/moritavakoli/hotel-booking

Sommario

Introduzione

La gestione efficiente delle prenotazioni riveste un ruolo fondamentale per il successo della gestione degli hotel. Un'accurata osservazione dei dati delle prenotazioni può aiutare ad ottimizzare l'occupazione delle stanze e pianificare in maniera ottimale le risorse in base al periodo.

Lo scopo principale di questo progetto è di utilizzare algoritmi di machine learning per il controllo delle prenotazioni al fine di migliorare l'efficienza nella gestione degli hotel. Utilizzando algoritmi di apprendimento automatico si mira a prevedere la cancellazione delle prenotazioni, per questo scopo si è ritenuto opportuno rappresentare i dati in una Knowledge Base al fine di estrarre nuove informazioni utili dal dataset originale.

Elenco argomenti di interesse

- Rappresentazione e ragionamento relazionale: partendo dai dati contenuti nel dataset selezionato ho utilizzato il linguaggio Prolog, lavorando quindi su una base di conoscenza, per ottenere nuove informazioni.
- Apprendimento supervisionato: ho selezionato algoritmi di machine learning utili ai fini della previsione quali KNN, Decision Tree, Random Forest, Ada Boost e Gradient Boosting.

Preprocessing

Il dataset utilizzato contiene le informazioni sulle prenotazioni in un city hotel ed un resort hotel di vario genere: quando è stata effettuata la prenotazione, quanto tempo è rimasta in attesa di essere confermata, numero di adulti, bambini e/o neonati, numero di richieste speciali, tipo di stanza richiesta, ma anche informazioni personali dei clienti come nome, mail, carta di credito e numero di cellulare. (Ovviamente le informazioni personali sono state oscurate)

Per prima cosa sono stati aggiunti un ID di prenotazione che risultava assente e un ID per cliente utile per una selezione più efficiente e semplice rispetto all'utilizzo del nome o della mail.

Inoltre, le features stays_in_weekend_nights e stays_in_week_nights sono state riunite in un'unica feature che indicasse il numero totale di notti per praticità quale total_nights_stay, e si è preferito sostituire la features categorica che indica il mese di prenotazione con una feature numerica che ne indicasse il numero.

Successivamente è stata svolta una indagine volta ad individuare i valori nulli, quindi è seguita l'eliminazione delle colonne che presentano molti valori nulli e di quelle che non sembrano avere un reale valore informativo utile al caso descritto. Si è ritenuto utile mantenere tra le colonne contenenti valori nulli children sostituendoli con 0 ed eliminare le prenotazioni del dataset che presentavano 0 ospiti o che avevano una durata inferiore a 1 notte.

Successivamente il dataset originale è stato convertito in lettere tutte minuscole, sono stati eliminati gli spazi e sostituiti con il simbolo "_" e convertiti i valori di children in interi poiché risultavano decimali e le varie features sono state suddivise in due file separati per comodità e facilità di interrogazione del dataset.

nome_dataset	features categoriche	features numeriche
customers_complete.csv	reservation_ID, customer_ID, customer_type, is_repeated_guest	previous_cancellations, adults, children, babies
reservations_complete.csv	reservation_ID, hotel, is_canceled, reserved_room_type,	lead_time, arrival_date_year, arrival_date_month, arrival_date_day_of_month, total_nights_stay,

	assigned_room_type, deposit_type, meal	booking_changes, days_in_waiting_list, adr, total_of_special_requests, reservation_status_date

I file si chiamano customers_complete e reservations_complete perché esistono già un file reservations ed uno customers dalle dimensioni molto ridotte rispetto ai complete, utilizzati per effettuare delle interrogazioni alla base di conoscenza che restituissero valori limite (max e min) e valori medi. Le features sono le stesse per entrambe le coppie di file. Questi dati sono stati mantenuti nel progetto per puro scopo illustrativo ma non sono stati utilizzati nella fase di predizione poiché risultava esclusa la maggior parte dei dati del dataset.

Rappresentazione e ragionamento relazionale

È stata creata una knowledge base utilizzando il linguaggio Prolog, in particolare la libreria pyswip che fornisce un'interfaccia di accesso a SWI-Prolog programmando in Python.

Sono stati definiti due individui, ognuno identificato dal proprio ID: reservation e customer. Tra le proprietà di ogni reservation c'è il codice cliente che permette di fare riferimento, tra i fatti definiti per l'individuo customer, al cliente di cui si desidera avere informazioni in più.

has_reservations(C, Res) :- findall(R, customer(R, C, _, _, _, _, _, _), Res)

restituisce la lista delle prenotazioni effettuate da un cliente

customer(Res, C) :- customer(Res, C, _, _, _, _, _, _)

restituisce il codice del cliente che ha effettuato la prenotazione data in input

una prenotazione ha un solo cliente di riferimento ma un cliente può aver effettuato più prenotazioni

Proprietà

Le proprietà associate agli individui sono le feature presenti nei dataset corrispondenti.

- Proprietà di un customer legate ad una sua prenotazione x:
  - babies(C, R, B)
  - children(C, R, C)
  - customer_type(C, R, T)
  - previous_cancellations(C, R, PC)
  - adults(C, R, A)
  - is_repeated_guest(C, R, RG)
- Proprietà di una reservation:
  - hotel(R, H)
  - is_canceled(R, C)
  - lead_time(R, LT)
  - arrival_date_year(R, Y)
  - arrival_date_month(R, M)
  - arrival_date_day_of_month(R, D)
  - total_nights(R, N)
  - reserved_room_type(R, RR)

- assigned_room_type(R, AR)
- booking_changes(R, C)
- deposit_type(R, DT)
- days_in_waiting_list(R, W)
- adr(R, A)
- total_of_special_requests(R, SR)
- reservation_status_date(R, RSD)
- meal(R, M)

Clausole

Al fine di ottenere nuove informazioni a partire dal dataset sono state selezionate delle clausole definite utili a identificare alcune caratteristiche nascoste delle prenotazioni:

- Si è ritenuto opportuno guardare alle tempistiche di una prenotazione: quanto tempo è rimasta in attesa di conferma? qual è la durata del soggiorno? la prenotazione è stata effettuata in anticipo o all'ultimo minuto?

```
% regola per definire se una prenotazione è di corta, media o lunga durata
short_stay(ReservationID) :- reservation(ReservationID,_,_,_,_,_,_,_,Nights,_,_,_,_,_,_,_,_,_,_), Nights < 3.
medium_stay(ReservationID) :- reservation(ReservationID,_,_,_,_,_,_,_,Nights,_,_,_,_,_,_,_,_,_,_), Nights >= 3, Nights < 7.
long_stay(ReservationID) :- reservation(ReservationID,_,_,_,_,_,_,_,Nights,_,_,_,_,_,_,_,_,_,_), Nights >= 7.
```

- Sono state osservate altre caratteristiche al fine di capire il carattere della prenotazione: è solo un weekend? per caso un weekend romantico? è una vacanza tra amici? si tratta di una vacanza per cui il cliente non ha badato a spese?

```
% regola per stabilire il giorno della settimana occupato in quella data in base alla formula Zellers Congruence
day_of_the_week(Year, Month, Day, DayOfWeek) :- Month > 2,  % Gennaio e febbraio devono essere considerati come i mesi 13 e 14 anno precedente
                                                NewMonth is Month - 2,
                                                NewYear is Year,
                                                Z is NewYear // 100,
                                                C is NewYear mod 100,
                                                F is Day + ((13 * NewMonth - 1) // 5) + C + (C // 4) + (Z // 4) - (2 * Z),
                                                DayOfWeek is (F mod 7 + 6) mod 7 + 1.

% regola che stabilisce se si tratta di una prenotazione per un weekend
weekend_booking(ReservationID) :-   reservation(ReservationID,_,_,_,_,_,_,_,Nights,_,_,_,_,_,_,_,_,_,_), Nights > 0, Nights < 3,
                                    reservation(ReservationID,_,_,_,_,Year,Month,Day,_,_,_,_,_,_,_,_,_,_,_),
                                    day_of_the_week(Year, Month, Day, DayOfWeek),
                                    member(DayOfWeek, [5, 6]). % Venerdì (5) o sabato (6)

% regola che stabilisce se potrebbe trattarsi di una prenotazione per un weekend romantico
romantic_weekend(ReservationID) :-  weekend_booking(ReservationID),
                                    customer(ReservationID,_,_,_,_,Adults,_,_,_), Adults = 2,
                                    customer(ReservationID,_,_,_,_,_,Children,_,_), Children = 0,
                                    customer(ReservationID,_,_,_,_,_,_,Babies,_), Babies = 0.

% regola che stabilisce se potrebbe trattarsi di una prenotazione per un viaggio di famiglia
family_vacation(ReservationID) :-   customer(ReservationID,_,_,_,_,Adults,_,_,_), Adults > 0,
                                    customer(ReservationID,_,_,_,_,_,Children,_,_), Children > 0,
                                    customer(ReservationID,_,_,_,_,_,_,Babies,_), Babies > 0.
```

nb. day_of_the_week è una regola ausiliaria creata per essere utilizzata nella regola weekend_booking a sua volta utile per rilevare se la prenotazione x è un weekend romantico, in particolare day_of_the_week usa una formula per riconoscere il giorno della settimana in base ad una data fornita

- Infine, sono state inserite una clausola volta a valutare se la stanza assegnata al cliente sia quella da lui richiesta in fase di prenotazione e una volta a valutare se possa aver fatto lui la richiesta di cambiare stanza

```
% regola per stabilire se la stanza assegnata è uguale a quella richiesta
changed_room_type(ReservationID) :- reservation(ReservationID,_,_,_,_,_,_,_,ReservedType,_,_,_,_,_,_,_,_),
                                    reservation(ReservationID,_,_,_,_,_,_,_,_,AssignedType,_,_,_,_,_,_,_), ReservedType \= AssignedType.

% regola per capire se il cambio di stanza è stato richiesto dal cliente
customer_requested_room_change(ReservationID) :-    changed_room_type(ReservationID),
                                                    reservation(ReservationID,_,_,_,_,_,_,_,_,_,_,Booking_changes,_,_,_,_,_), Booking_changes > 0;
                                                    reservation(ReservationID,_,_,_,_,_,_,_,_,_,_,_,_,_,_,Special_requests,_,_), Special_requests > 0.
```

nb. per ogni clausola sono state stabilite delle regole arbitrarie utili a definire la prenotazione, non sono state riportate tutte le clausole scelte per il progetto poiché simili nella forma ad altre già descritte

Sono state definite delle clausole, come accennato a pagina 3, che calcola i valori medi, massimi o minimi per alcune delle proprietà numeriche. Sfortunatamente al fine di realizzare questo tipo di indagine il dataset è stato ridotto di molto in quanto le risposte ottenute dalle query occupavano uno spazio di memoria superiore a quello consentito.

Una volta ridotto il dataset è stato possibile visualizzare correttamente i risultati delle query, riporto un esempio per tipo (una media, un massimo e un minimo):

```
# numero medio di notti prenotate
prolog.assertz('avg_stay_in(Avg) :- findall(N, reservation(_,_,_,_,_,_,_,N,_,_,_,_,_,_,_,_,_), Values), sumlist(Values, Sum), length(Values, Count), Avg is Sum/Count')
# stanza più richiesta - viene restituito solo il primo tipo se ci sono più stanze con la stessa richiesta
prolog.assertz('most_desired_room(Room_type) :-  findall(RR, reservation(_,_,_,_,_,_,_,_,RR,_,_,_,_,_,_,_,_), Room_list), bagof(Oc-Room, (member(Room, Room_list), findall(X, member(X, Room_list), L), length(L, Oc)), Room_types_oc), keysort(Room_types_oc, Ordered_room_types_oc), reverse(Ordered_room_types_oc, Ordered_room_types_oc_dec), Ordered_room_types_oc_dec = [_-Room_type | _]')
```

```
# stanza meno richiesta - viene restituito solo il primo tipo se ci sono più stanze con la stessa richiesta
prolog.assertz('least_desired_room(Room_type) :-  findall(RR, reservation(_,_,_,_,_,_,_,_,RR,_,_,_,_,_,_,_,_), Room_list), bagof(Oc-Room, (member(Room, Room_list), findall(X, member(X, Room_list), L), length(L, Oc)), Room_types_oc), keysort(Room_types_oc, Ordered_room_types_oc), Ordered_room_types_oc = [_-Room_type | _]')
```

- *findall(Template, Goal, List) è un predicato che crea una lista di istanze di Template che soddisfano il Goal. Esamina tutte le possibili soluzioni e restituisce una lista contenente tutte le istanze soddisfatte.*
- *bagof(Template, Goal, Bag) è simile a findall/3, ma raggruppa le soluzioni in sacche separate in base alle istanze delle variabili non vincolate nel Template. Se una variabile non vincolata nel Template ha più soluzioni, le soluzioni verranno raggruppate in sacche separate.*
- *sum_list(List, Sum) è un predicato che calcola la somma degli elementi nella lista List e restituisce il risultato in Sum. La lista List deve contenere solo numeri.*

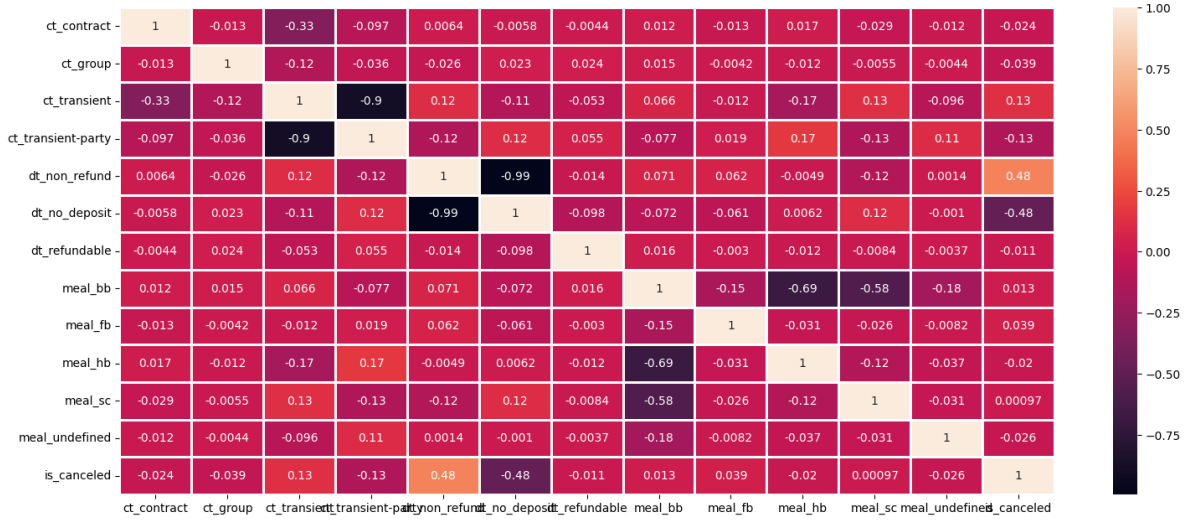
I risultati delle query appena descritte sono stati inseriti nei corrispettivi file commands_c per i clienti e commands_r per le prenotazioni.

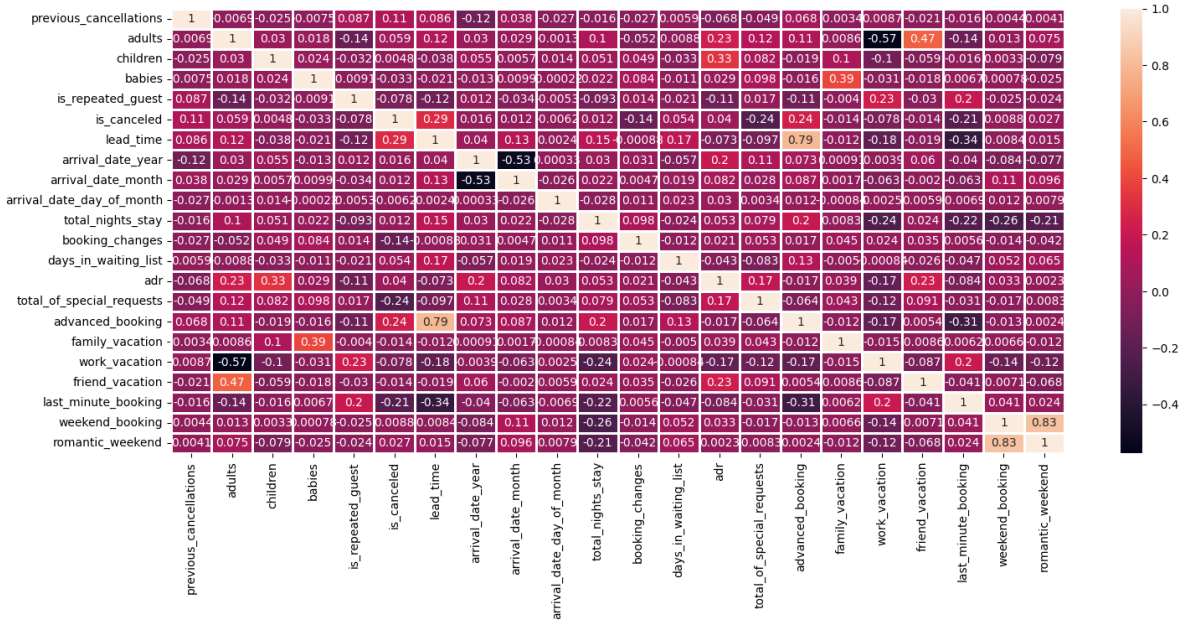
Apprendimento supervisionato

Una volta aggiunte le features ingegnerizzate al dataset originale, a seguito di un ulteriore preprocessing dei dati che sarà descritto nel paragrafo successivo, sono stati utilizzati diversi modelli di classificazione ed in varie modalità al fine di individuare il modello migliore per il caso d'uso scelto tra KNN, Decision Tree, Random Tree, Ada Boost e Gradient Boosting.

Preprocessing

Osserviamo la correlazione tra le features:





In particolare, guardiamo la correlazione con l'oggetto target is_canceled:

```
dt_no_deposit                 0.477531
lead_time                     0.291670
changed_room_type             0.245615
advanced_booking              0.242297
total_of_special_requests     0.235994
last_minute_booking           0.211149
booking_changes               0.144750
city_hotel                    0.135358
resort_hotel                  0.135358
ct_transient                  0.134447
ct_transient-party            0.125428
status_update_year            0.118420
previous_cancellations        0.110003
work_vacation                 0.077751
is_repeated_guest             0.077639
status_update_month           0.068108
adults                        0.058847
days_in_waiting_list          0.053806
adr                           0.040044
ct_group                      0.038946
meal_fb                       0.038678
status_update_day             0.034279
babies                        0.032621
romantic_weekend              0.027211
meal_undefined                0.025854
ct_contract                   0.023916
meal_hb                       0.019922
arrival_date_year             0.015507
family_vacation               0.014224
friend_vacation               0.013975
meal_bb                       0.013398
total_nights_stay             0.012495
arrival_date_month            0.011752
dt_refundable                 0.011482
weekend_booking               0.008833
arrival_date_day_of_month     0.006204
children                      0.004804
meal_sc                       0.000970
Name: is_canceled, dtype: float64
```

Dopo aver combinato i due dataset in cui era stato separato l'originale, sono state apportate delle modifiche al dataset:

- le features reserved_room_type e assigned_room_type sono state eliminate poichè rappresentate solo tramite sigle non indicative, non hanno potere informativo, l'informazione importante però, ovvero la differenza tra le due features è rimasta grazie all'introduzione nel dataset della feature changed_room_type
- tra le feature ingegnerizzate, alcune ritenute più rilevanti sono state inserite all'interno del dataset: changed_room_type, advanced_booking, family_vacation, work_vacation, friend_vacation, last_minute_booking, weekend_booking, romantic_weekend (luxury_booking

ad esempio non ha prodotto risultati e quindi non è stata inserita, altre avrebbero ulteriormente appesantito il dataset per cui è stata fatta una selezione)

- gli Id sono stati eliminati al fine di evitare leakage, non hanno inoltre potere informativo
- la feature reservation_status_date è stata divisa in tre feature corrispondenti ad anno, mese e giorno per ciascuna data per avere informazioni temporali più dettagliate e identificare meglio eventuali pattern

passando ad ulteriori modifiche sulla forma invece:

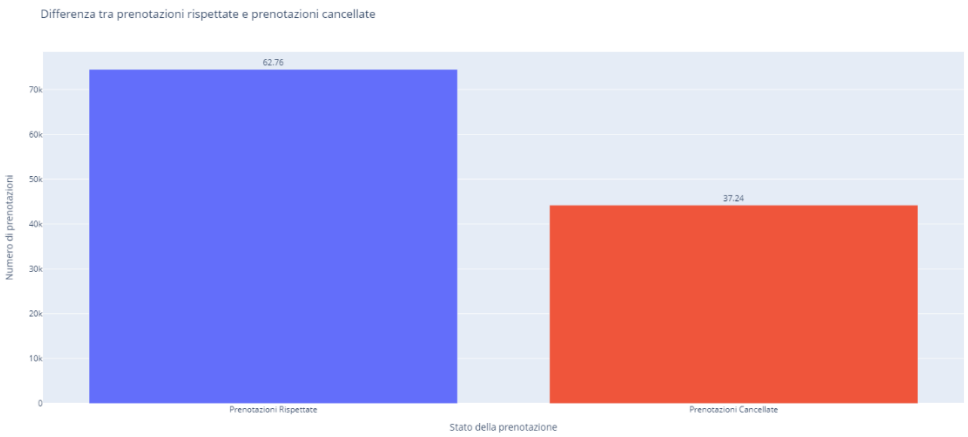
- i valori contenuti nelle variabili categoriche non booleane sono stati codificati in variabili booleane indipendenti, è stata quindi applicata la metodologia one hot encoding: il KNN calcola la distanza tra le osservazioni del dataset, di conseguenza utilizzando una mappatura numerica potrebbe essere assegnata una relazione d'ordine arbitraria ai numeri che potrebbe influire negativamente sulle distanze calcolate, per quanto riguarda invece gli altri algoritmi, questi sono in grado di gestire autonomamente la mappatura delle features categoriche ma si proporrebbe comunque il problema della relazione d'ordine arbitraria nel caso in cui tra le categorie non esistesse una relazione d'ordine naturale
- infine, i valori del dataset sono stati normalizzati tramite Min-Max Scaler, il KNN infatti richiede che le osservazioni abbiano una scala simile per ottenere risultati ottimali
  nb. il Min-Max Scaler preserva la relazione d'ordine tra le osservazioni, quindi anche dopo la normalizzazione la relazione tra queste resta la stessa

Osservazioni sui dati

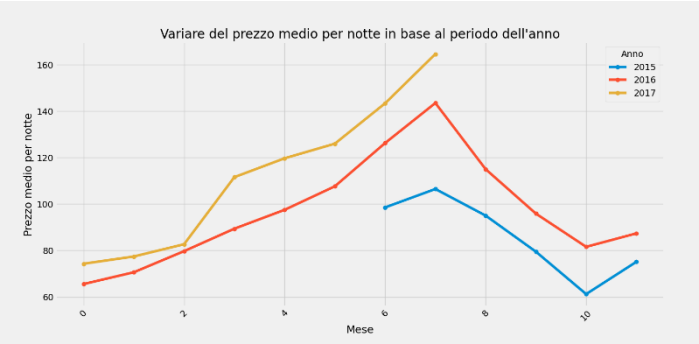
Diamo uno sguardo generale al dataset:

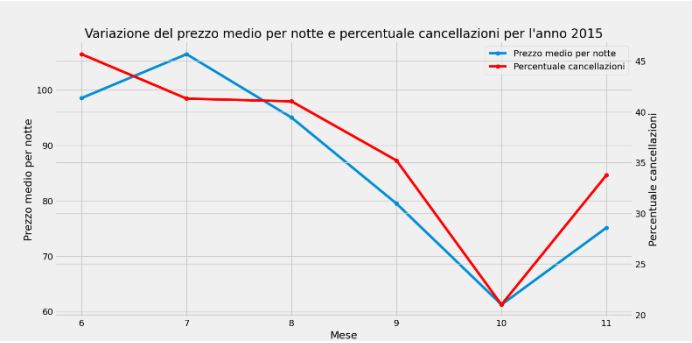
```
RangeIndex: 118675 entries, 0 to 118674
Data columns (total 30 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   reservation_ID               118675 non-null  object
 1   customer_ID                  118675 non-null  object
 2   previous_cancellations       118675 non-null  int64
 3   customer_type                118675 non-null  object
 4   adults                       118675 non-null  int64
 5   children                     118675 non-null  int64
 6   babies                       118675 non-null  int64
 7   is_repeated_guest            118675 non-null  int64
 8   hotel                        118675 non-null  object
 9   is_canceled                  118675 non-null  int64
 10  lead_time                    118675 non-null  int64
 11  arrival_date_year            118675 non-null  int64
 12  arrival_date_month           118675 non-null  int64
 13  arrival_date_day_of_month    118675 non-null  int64
 14  total_nights_stay            118675 non-null  int64
 15  booking_changes              118675 non-null  int64
 16  deposit_type                 118675 non-null  object
 17  days_in_waiting_list         118675 non-null  int64
 18  adr                          118675 non-null  float64
 19  total_of_special_requests    118675 non-null  int64
 20  reservation_status_date      118675 non-null  object
 21  meal                         118675 non-null  object
 22  changed_room_type            118675 non-null  int64
 23  advanced_booking             118675 non-null  int64
 24  family_vacation              118675 non-null  int64
 25  work_vacation                118675 non-null  int64
 26  friend_vacation              118675 non-null  int64
 27  last_minute_booking          118675 non-null  int64
 28  weekend_booking              118675 non-null  int64
 29  romantic_weekend             118675 non-null  int64
dtypes: float64(1), int64(22), object(7)
memory usage: 27.2+ MB
```

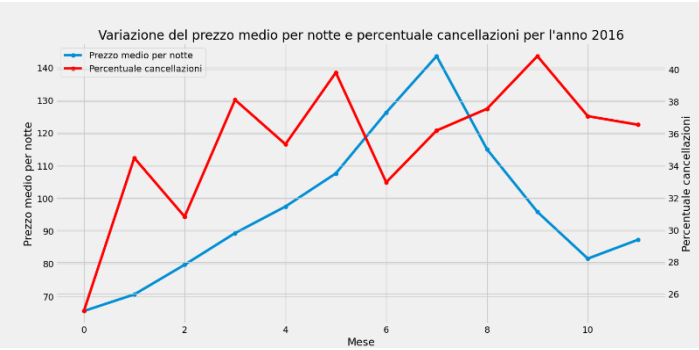
Per prima cosa possiamo osservare il rapporto tra il numero di prenotazioni onorate e quello di prenotazioni cancellate.

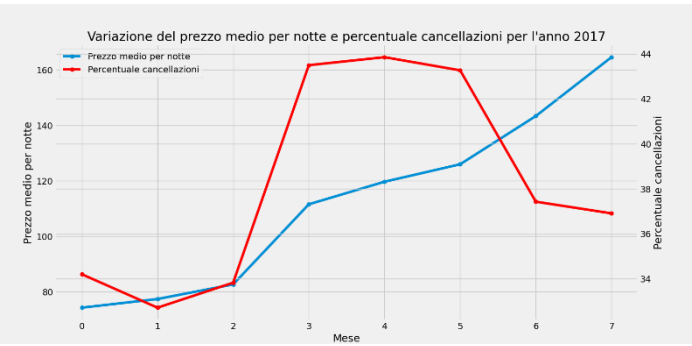


Successivamente possiamo osservare se e come il prezzo medio per notte di una camera influenzi il numero di cancellazioni.

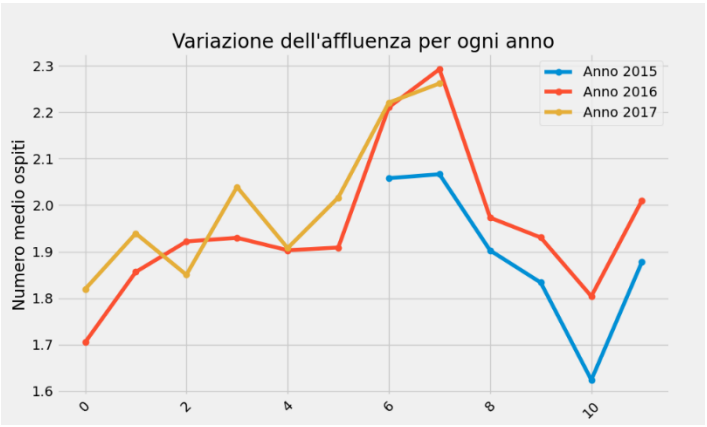


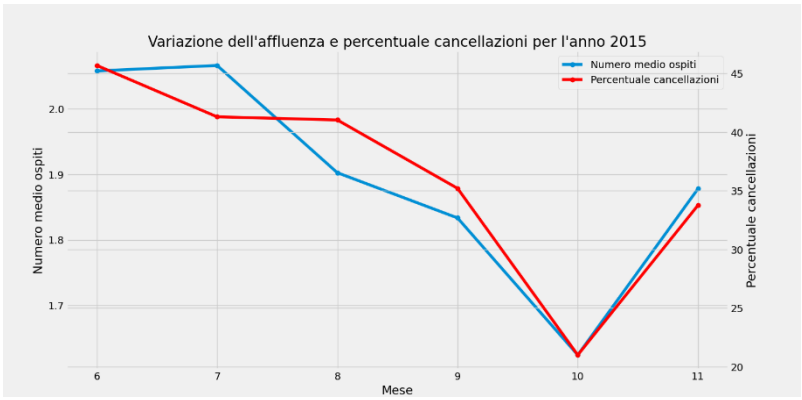


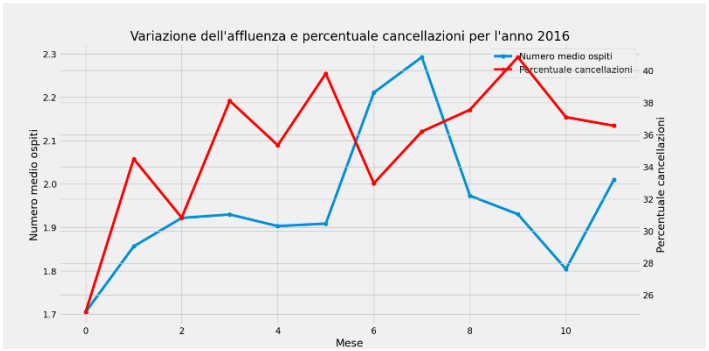




Possiamo anche osservare il rapporto tra affluenza e cancellazioni.







Modelli[1]

In un primo momento tutti i modelli sono stati testati senza effettuare modifiche ai parametri in quanto si voleva eseguire un primo tentativo per capire in che direzione procedere. I risultati sono stati sin da subito positivi:

```
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.99      0.94     14793
           1       0.97      0.83      0.90      8942

    accuracy                           0.93     23735
   macro avg       0.94      0.91      0.92     23735
weighted avg       0.93      0.93      0.93     23735

F2 Score: 0.8548781049817965

Confusion Matrix :
[[14583   210]
 [ 1522  7420]]

Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     14793
           1       0.95      0.94      0.95      8942

    accuracy                           0.96     23735
   macro avg       0.96      0.96      0.96     23735
weighted avg       0.96      0.96      0.96     23735

F2 Score: 0.9444121071012808

Confusion Matrix :
[[14330   463]
 [  505  8437]]

Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     14793
           1       0.98      0.88      0.93      8942

    accuracy                           0.95     23735
   macro avg       0.96      0.94      0.94     23735
weighted avg       0.95      0.95      0.95     23735

F2 Score: 0.8979311921362443
Confusion Matrix :
[[14672   121]
 [ 1086  7856]]
```

```
Ada Boost Classifier Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     14793
           1       0.95      0.94      0.95      8942

    accuracy                           0.96     23735
   macro avg       0.96      0.96      0.96     23735
weighted avg       0.96      0.96      0.96     23735

F2 Score: 0.9447814311614476
Confusion Matrix :
[[14326   467]
 [  500  8442]]

Gradient Boosting Classifier Classification Report:
              precision    recall  f1-score   support

           0       0.87      1.00      0.93     14793
           1       0.99      0.76      0.86      8942

    accuracy                           0.91     23735
   macro avg       0.93      0.88      0.90     23735
weighted avg       0.92      0.91      0.91     23735

F2 Score: 0.8006566604127582
Confusion Matrix :
[[14749    44]
 [ 2114  6828]]
```

[1] *nb. tutti i modelli sono stati valutati su accuracy, precision, recall e F2 score, le prime per avere un giudizio più completo sui modelli, la F2 score in quanto il fine ultimo del progetto è quello di riuscire ad individuare quali saranno le prenotazioni cancellate in quanto, non solo è possibile notare uno sbilanciamento nel dataset rispetto alle prenotazioni cancellate e onorate, ma la mancata occupazione delle stanze causa una perdita di denaro e per questo si vogliono minimizzare i falsi negativi (→ recall), allo stesso tempo non si vuole rischiare che si verifichino troppi inconvenienti causati da una cattiva gestione dell'overbooking (→ anche precision è importante), di conseguenza la scelta ultima del modello si baserà soprattutto sul suo valore*

Analizzando i risultati della classificazione per i diversi modelli, possiamo trarre le seguenti conclusioni:

- KNN mostra una buona capacità di classificazione complessiva con un'accuratezza del 93%. Il modello ottiene una F2 score di 0.8548781049817965, indicando che fornisce una buona combinazione di precisione e recall, anche se con una recall leggermente inferiore per la classe positiva.
- Decision Tree ha una F2 score di 0.9444121071012808, che indica un'ottima capacità del modello di rilevare correttamente gli esempi positivi, insieme a un'alta precisione generale. Questo modello mostra buone prestazioni in termini di bilanciamento tra recall e precisione e raggiunge un'accuratezza del 96%.
- Random Forest ha un'accuratezza del 95% (con una buona capacità di classificazione per entrambe le classi) e una F2 score di 0.8979311921362443, che rappresenta una buona combinazione di precisione e recall, anche se con una recall leggermente inferiore per la classe positiva rispetto al Decision Tree.
- Ada Boost Classifier ottiene una F2 score di 0.9447814311614476, di poco superiore a quella del Decision Tree, mentre l'accuratezza è la stessa.
- Gradient Boosting Classifier ha una F2 score di 0.8006566604127582, che indica che il modello ha una maggiore enfasi sulla precisione rispetto alla recall, che risulta essere relativamente bassa per la classe positiva, influenzando l'F2 score complessivo. L'accuratezza in questo caso è del 91%.

Complessivamente, i modelli Decision Tree, Random Forest e Ada Boost Classifier mostrano prestazioni ottime, ma è l'Ada Boost Classifier che raggiunge la migliore combinazione di accuratezza, precisione e recall.

In ogni caso, al fine di avere una valutazione più accurata dei modelli, è stato utilizzato successivamente lo schema 10-fold cross-validation, infatti la cross validation suddivide il dataset in diverse parti e addestra/valuta il modello su diverse combinazioni di training/validation set, fornendo una stima più affidabile delle prestazioni generali del modello.

Anche in questo caso i risultati ottenuti sono stati ottimi:

```
KNN: Average Accuracy Score: 0.9251737939751422, Average Precision Score: 0.9665608252153219,
    Average Recall Score: 0.8270934803278168, Average F2 Score: 0.8516683272837989

Decision Tree: Average Accuracy Score: 0.9561407204550243, Average Precision Score: 0.9426392312263214,
            Average Recall Score: 0.9389694323929622, Average F2 Score: 0.9396963764616204

Random Forest: Average Accuracy Score: 0.9499684010954287, Average Precision Score: 0.9866002767938836,
            Average Recall Score: 0.8771967739769313, Average F2 Score: 0.8970846895444671

Ada Boost Classifier: Average Accuracy Score: 0.9563303138824522, Average Precision Score: 0.9433544556170359,
                    Average Recall Score: 0.9387071193728966, Average F2 Score: 0.9396260678693176

Gradient Boosting Classifier: Average Accuracy Score: 0.910048451653676, Average Precision Score: 0.9917312285841744,
                            Average Recall Score: 0.7641464756627869, Average F2 Score: 0.8008768395127801
```

Dal confronto dei risultati, il modello Decision Tree ha ottenuto l'F2 score più alto con un valore di 0.9396963764616204, è quindi particolarmente efficace nel gestire l'obiettivo che ci si è proposti di realizzare con questo progetto e presenta anche un'ottima accuracy (di poco inferiore a quella del suo secondo, l'Ada Boost).

KNN, Random Forest e Ada Boost Classifier hanno ottenuto punteggi rispettivamente di 0.8516683272837989, 0.8970846895444671 e 0.9396260678693176. Questi modelli sono tutti validi, ma non raggiungono il punteggio del Decision Tree.

Il Gradient Boosting Classifier ha ottenuto l'F2 score più basso, pari a 0.8008768395127801. Nonostante abbia una precisione molto alta (0.9917312285841744), la recall è relativamente bassa (0.7641464756627869).

Confrontando i risultati ottenuti con un semplice addestramento sul training set e quelli ottenuti con la metodologia 10-fold cross-validation, possiamo notare le seguenti differenze:

- In termini di accuracy, i modelli addestrati con la metodologia 10-fold cross-validation mantengono prestazioni simili rispetto ai risultati ottenuti con un semplice addestramento sul training set. Questo suggerisce che i modelli sono generalizzabili e non soffrono di overfitting.
- I punteggi di precision, recall e F2 score ottenuti con la cross-validation sono più affidabili in quanto tengono conto delle variazioni nei dati di addestramento e di validazione. Questi punteggi forniscono una valutazione più accurata delle capacità del modello nel gestire i dati non visti durante l'addestramento.
- In generale, i modelli presentano una consistenza nelle prestazioni tra l'addestramento sul training set e la cross-validation, quindi non sono particolarmente influenzati dalla

suddivisione dei dati durante la validazione incrociata e le loro prestazioni sono stabili e coerenti.

Decision Tree e Ada Boost restano in entrambi i casi i modelli migliori, uno in vantaggio con la prima valutazione, l'altro in vantaggio con la seconda, di conseguenza si è ritenuto opportuno cercare di selezionare i parametri migliori per ogni modello, sia per migliorare ulteriormente le prestazioni sia per giungere ad una decisione finale sul migliore modello da utilizzare.

I parametri migliori sono stati ottenuti tramite l'applicazione dell'algoritmo Grid Search utilizzando come metrica la F2 Score e, per ogni tentativo effettuato combinando i diversi parametri, è stata effettuata la cross-validation. Di seguito, è fornita una descrizione più dettagliata di come i parametri sono stati selezionati per ciascun modello:

- Per il modello K-Nearest Neighbors (KNN), il parametro selezionato per la Grid Search è il numero di vicini (n_neighbors). Nel caso specifico, sono stati testati valori compresi tra 3 e 10. È importante trovare un equilibrio tra un numero sufficiente di vicini per ottenere una buona precisione e un numero troppo alto che potrebbe ridurre la sensibilità del modello. Dato il dataset relativamente grande, valori relativamente bassi sono stati scelti per ottenere una buona precisione senza complicare eccessivamente il modello, nonostante un numero più alto di vicini possa ridurre il rischio di overfitting.
- Per il Decision Tree Classifier (DTC), il parametro selezionato per la Grid Search è la massima profondità dell'albero (max_depth). È stato scelto di testare valori compresi tra 3 e 10. La scelta della massima profondità mira a controllare il livello di complessità dell'albero e a prevenire l'overfitting. Limitando il range dei valori di max_depth, si cerca di trovare un compromesso tra la capacità del modello di adattarsi ai dati e la sua capacità di generalizzare su nuovi dati.
- Per il Random Forest Classifier (RFC), sono stati selezionati due parametri per la Grid Search: il numero di alberi (n_estimators) e la massima profondità degli alberi (max_depth). Per n_estimators, sono stati testati valori compresi tra 50 e 200, mentre per max_depth sono stati testati valori compresi tra 5 e 20. L'obiettivo è bilanciare la complessità del modello con la sua capacità di generalizzare su nuovi dati.
- Per l'Ada Boost Classifier (ABC), i parametri selezionati per la Grid Search sono il numero di stimatori (n_estimators) e il learning rate. Sono stati testati valori compresi tra 50 e 200 per il numero di stimatori al fine di valutare come il modello si comporta con diversi numeri.

Testare valori compresi tra 50 e 200 per gli stimatori consente di esplorare un intervallo ampio e migliorare la capacità di predizione del modello senza rendere il modello troppo complesso o richiedere troppo tempo di addestramento, mentre valori più bassi per il learning rate riducono l'influenza di ciascun stimatore richiedendo però un maggior numero di stimatori per ottenere una buona accuratezza. L'obiettivo è sempre quello di dare stabilità e generalizzare il modello.

- Infine, per il Gradient Boosting Classifier (GBC), sono stati selezionati due parametri per la Grid Search: il numero di stimatori (n_estimators) e il learning rate. I valori testati per n_estimators sono compresi tra 50 e 200, mentre per learning_rate sono 0.01, 0.1 e 1.0. Le scelte effettuate rispecchiano le motivazioni date per l'ABC.

Le valutazioni per tutte le combinazioni di parametri e per ogni modello si trovano nel file del progetto "valutazione_k_fold_&_grid_search.txt", in ogni caso guardiamo le valutazioni ottenute con i migliori parametri:

- KNN best parameters: {'n_neighbors': 3}:
  Average Accuracy Score: 0.9316, Average Precision Score: 0.9628, Average Recall Score: 0.8485, F2 Score: 0.8691.
- Decision Tree best parameters: {'max_depth': 10}:
  Average Accuracy Score: 0.8415, Average Precision Score: 0.8956, Average Recall Score: 0.6503, F2 Score: 0.6876.
- Random Forest best parameters: {'max_depth': 20, 'n_estimators': 150}:
  Average Accuracy Score: 0.9277, Average Precision Score: 0.9821, Average Recall Score: 0.8201, F2 Score: 0.8481.
- Ada Boost Classifier best parameters: {'learning_rate': 0.1, 'n_estimators': 50}:
  Average Accuracy Score: 0.9566, Average Precision Score: 0.9433, Average Recall Score: 0.9395, F2 Score: 0.9403.
- Gradient Boosting Classifier best parameters: {'learning_rate': 1.0, 'n_estimators': 200}:
  Average Accuracy Score: 0.9965, Average Precision Score: 0.9989, Average Recall Score: 0.9916, F2 Score: 0.9931.

Rispetto ai valori ottenuti precedentemente il miglioramento più significativo si è visto nel Gradient Boosting Classifier che raggiunge prestazioni eccellenti, KNN e Ada Boost vedono un leggero miglioramento, mentre il Random Forest un leggero peggioramento. Il Decision Tree che

precedentemente era stato valutato come uno dei migliori modelli, ha visto calare drasticamente le sue prestazioni che comunque sono buone prestazioni pur non essendo le migliori a confronto con quelle degli altri classificatori.

A questo punto i 5 modelli sono stati addestrati ognuno sul training set con i parametri risultati migliori e successivamente valutati sul test set. Ecco i risultati:

```
KNN
Confusion Matrix:
[[14531   262]
 [ 1363  7579]]
F2 Score: 0.8689720011924144
              precision    recall  f1-score   support

           0       0.91      0.98      0.95     14793
           1       0.97      0.85      0.90      8942

    accuracy                           0.93     23735
   macro avg       0.94      0.91      0.93     23735
weighted avg       0.93      0.93      0.93     23735

Decision Tree
Confusion Matrix:
[[13987   806]
 [ 3100  5842]]
F2 Score: 0.6886552244436062
              precision    recall  f1-score   support

           0       0.82      0.95      0.88     14793
           1       0.88      0.65      0.75      8942

    accuracy                           0.84     23735
   macro avg       0.85      0.80      0.81     23735
weighted avg       0.84      0.84      0.83     23735

Random Forest
Confusion Matrix:
[[14649   144]
 [ 1605  7337]]
F2 Score: 0.8482277046868136
              precision    recall  f1-score   support

           0       0.90      0.99      0.94     14793
           1       0.98      0.82      0.89      8942

    accuracy                           0.93     23735
   macro avg       0.94      0.91      0.92     23735
weighted avg       0.93      0.93      0.92     23735
```

```
Ada Boost Classifier
Confusion Matrix:
[[14320   473]
 [  503  8439]]
F2 Score: 0.9443822739480754
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     14793
           1       0.95      0.94      0.95      8942

    accuracy                           0.96     23735
   macro avg       0.96      0.96      0.96     23735
weighted avg       0.96      0.96      0.96     23735

Gradient Boosting Classifier
Confusion Matrix:
[[14788     5]
 [   63  8879]]
F2 Score: 0.9942443787512316
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     14793
           1       1.00      0.99      1.00      8942

    accuracy                           1.00     23735
   macro avg       1.00      1.00      1.00     23735
weighted avg       1.00      1.00      1.00     23735
```

Sui valori ottenuti dalla valutazione dei modelli addestrati con i parametri migliori e poi testati sul test set vediamo nella maggior parte dei casi un leggero miglioramento delle prestazioni, nel caso del KNN un leggero calo che potrebbe semplicemente essere dovuto al fatto che durante la k-fold cross-validation si tiene conto della variabilità dei dati e si cerca di trovare i parametri che funzionano bene in generale su diverse parti del dataset, questo però non garantisce che i parametri ottimali trovati tramite la cross-validation siano i migliori per il dataset specifico.

In generale, la minima differenza nelle prestazioni indica che il modello si sta comportando in maniera coerente.

In ogni caso, è stato ritenuto opportuno sperimentare ulteriormente con i parametri allargandone il range:

```
classifiers = [
    ('KNN', KNeighborsClassifier(), {'n_neighbors': range(3, 11)}),
    ('Decision Tree', DecisionTreeClassifier(), {'max_depth': range(3, 11)}),
    ('Random Forest', RandomForestClassifier(), {'n_estimators': range(50, 201, 50), 'max_depth': range(5, 21, 5)}),
    ('Ada Boost Classifier', AdaBoostClassifier(estimator=DecisionTreeClassifier()), {'n_estimators': range(50, 201, 50), 'learning_rate': [0.01, 0.1, 1.0]}),
    ('Gradient Boosting Classifier', GradientBoostingClassifier(), {'n_estimators': range(50, 201, 50), 'learning_rate': [0.01, 0.1, 1.0]})
]
```

```
classifiers = [
    ('KNN', KNeighborsClassifier(), {'n_neighbors': range(2, 11)}),
    ('Decision Tree', DecisionTreeClassifier(), {'max_depth': range(3, 21)}),
    ('Random Forest', RandomForestClassifier(), {'n_estimators': range(50, 301, 50), 'max_depth': range(5, 31, 5)}),
    ('Ada Boost Classifier', AdaBoostClassifier(estimator=DecisionTreeClassifier()), {'n_estimators': range(50, 301, 50), 'learning_rate': [0.01, 0.1, 1.0]}),
    ('Gradient Boosting Classifier', GradientBoostingClassifier(), {'n_estimators': range(50, 201, 50), 'learning_rate': [0.01, 0.1, 1.0]})
]
```

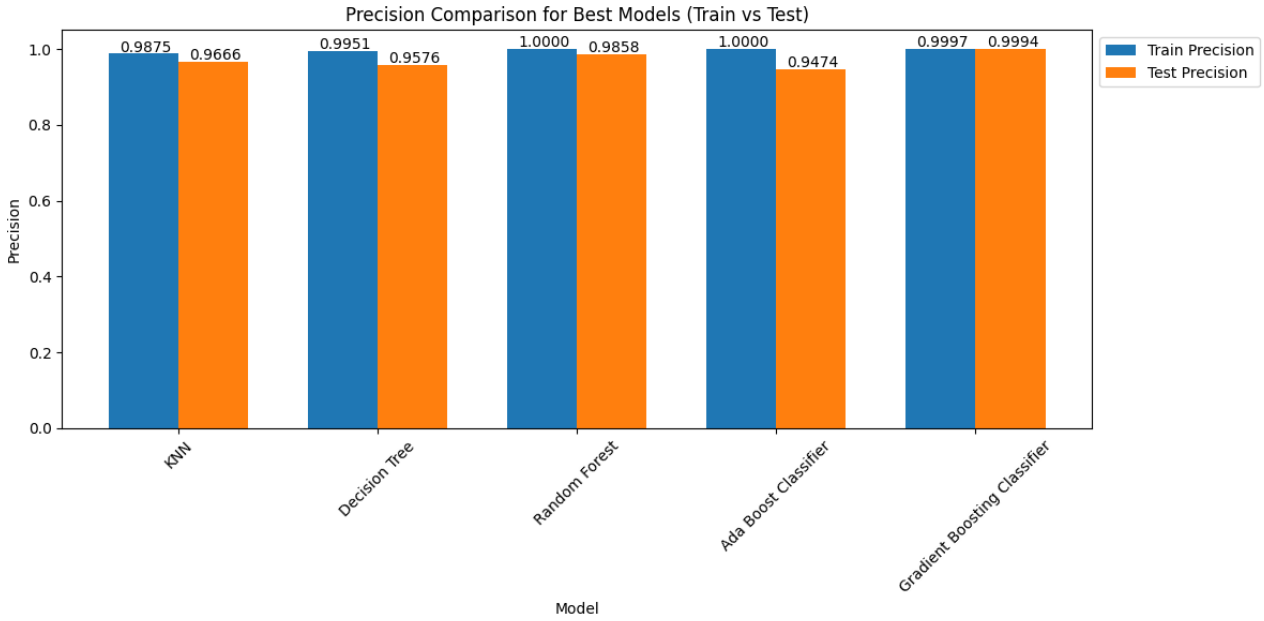
2

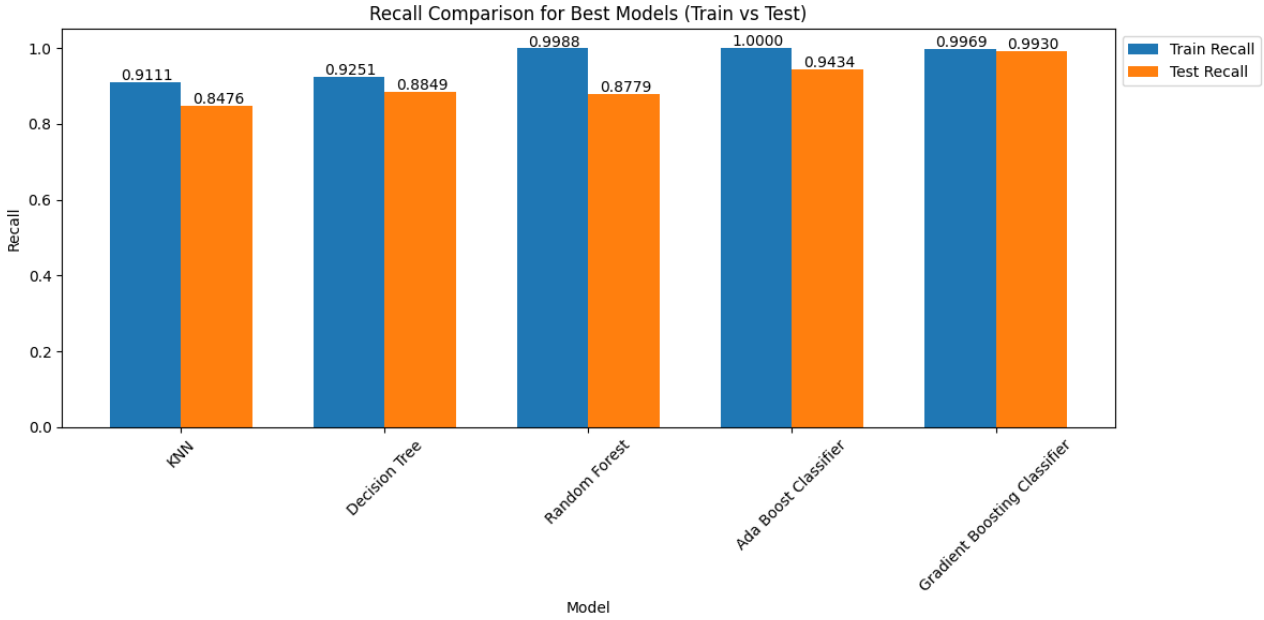
Con il nuovo range, i parametri migliori per ogni modello sono risultati essere:

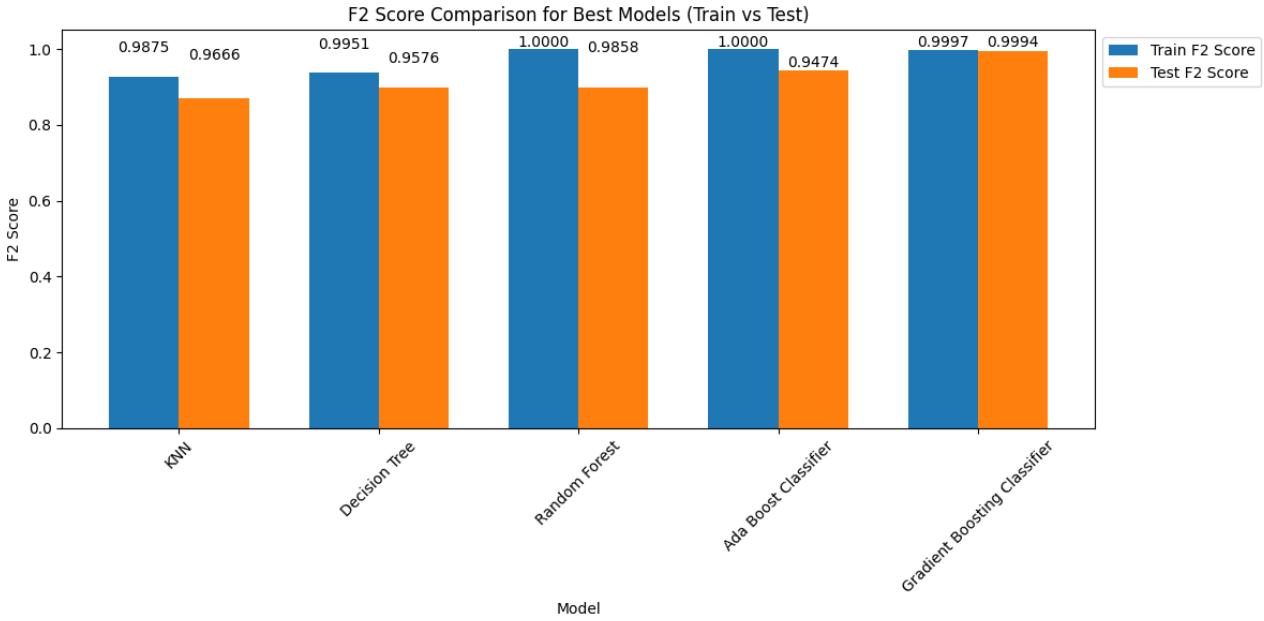
- Per il KNN, il numero di vicini migliore da considerare resta il 3.
- Per il Decision Tree la profondità che ha dato l'F2 migliore è stata 20:
  Average Accuracy Score: 0.9413, Average Precision Score: 0.9548, Average Recall Score: 0.8837, F2 Score: 0.8971.
- il Random Forest offre prestazioni migliori con una profondità di 30 e un numero di stimatori pari a 300: Average Accuracy Score: 0.9506, Average Precision Score: 0.9875, Average Recall Score: 0.8781, F2 Score: 0.8980.
- Nel caso dell'Ada Boost le prestazioni migliori si sono avute con un learning rate di 1.0 e un numero di stimatori pari a 300: Average Accuracy Score: 0.9565, Average Precision Score: 0.9431, Average Recall Score: 0.9396, F2 Score: 0.9403.
- Non si aggiungono considerazioni sul Gradient Boosting perché sono stati mantenuti gli stessi parametri.

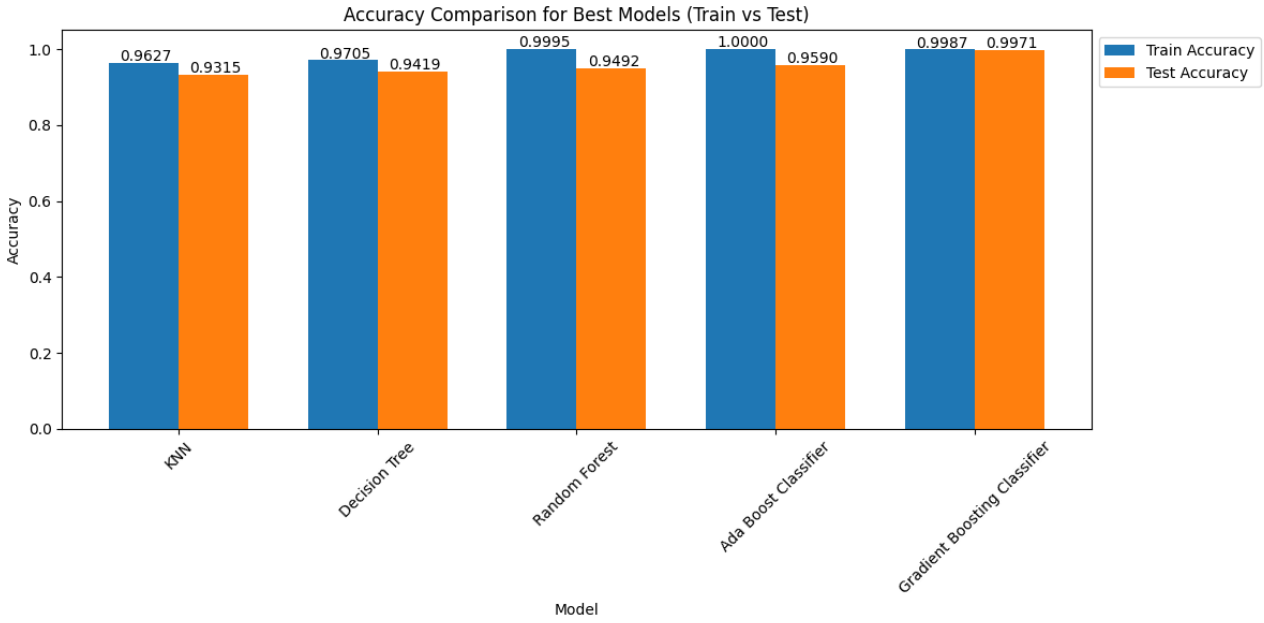
[2] i parametri del Gradient Boosting non sono stati modificati visto l'eccellente miglioramento del modello ottenuto già con i precedenti parametri

A questo punto i modelli sono stati addestrati sul training set e valutati sul test set utilizzando i parametri suggeriti dalla ricerca. Seguono dei grafici di confronto:









In conclusione, tutti i modelli hanno ottenuto prestazioni notevoli, con accuratezza e F2 score elevati. Tuttavia, c'è una leggera tendenza all'overfitting in KNN, Decision Tree e Random Forest, nell'Ada Boost non sembra esserci un problema di overfitting significativo.

Dai risultati ottenuti con il modello Gradient Boosting, possiamo osservare che il modello ha raggiunto un'elevata precisione sia nel training che nel test. La precisione sul training è del 99.87%, indicando che il modello è in grado di adattarsi molto bene ai dati di addestramento. Inoltre, l'F2 score sul training è del 99.97%, dimostrando che il modello riesce a identificare correttamente i casi positivi.

Nel caso del test, il modello ha ottenuto un'accuracy del 99.71% e un F2 score del 99.94%. Questi valori sono leggermente inferiori rispetto al training, ma rimangono comunque molto elevati. Indicano che il modello è in grado di generalizzare bene su nuovi dati e di mantenere alte prestazioni anche al di fuori del set di addestramento.

```
Gradient Boosting Classifier
Confusion Matrix:
[[14788     5]
 [   63  8879]]
F2 Score: 0.9942443787512316
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     14793
           1       1.00      0.99      1.00      8942

    accuracy                           1.00     23735
   macro avg       1.00      1.00      1.00     23735
weighted avg       1.00      1.00      1.00     23735
```