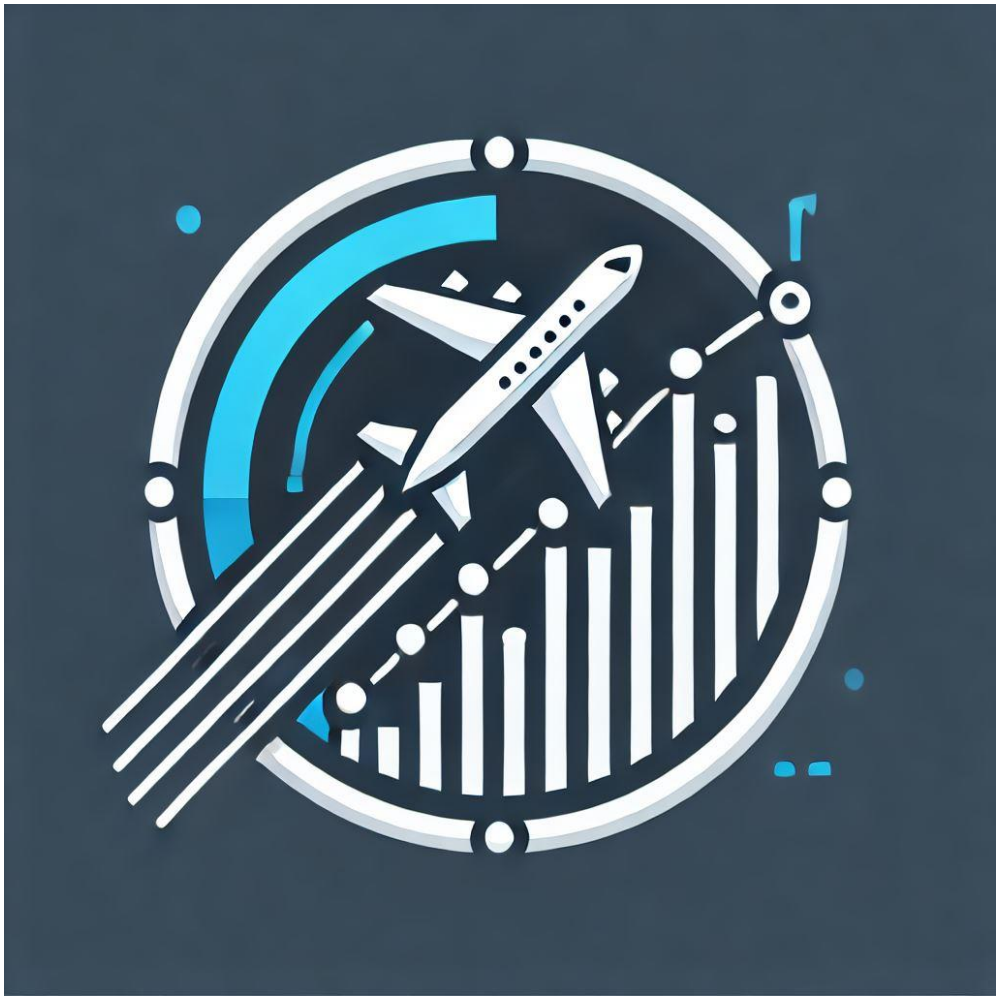
AirTips – Flight Fare Prediction



Lucageneroso Cammarota

mat: 0512116941

“Machine Learning”

Link al repository Github: https://github.com/lucageneroso/AirTip.git

Sommario

Problem statement

Il problema posto in essere prevede lo sviluppo di un modello di Machine Learning in grado di prevedere se il costo di un biglietto aereo è destinato a diminuire o meno. Si intende, cioè, sviluppare un modello adatto a consigliare all'utilizzatore se è il momento giusto per prenotare il volo o se aspettare è una scelta più saggia.

Esplorando le varie possibilità che oggi giorno ci sono su internet, ho individuato in AirHint un punto di riferimento.

Per sviluppare un modello di machine learning che preveda la convenienza di prenotare un biglietto aereo in un determinato momento, è fondamentale comprendere i fattori che influenzano le variazioni dei prezzi dei voli. AirHint, ad esempio, analizza le fluttuazioni dei prezzi per compagnia aerea, regione e rotta, fornendo raccomandazioni sul momento migliore per prenotare.

Nello specifico, ho ritenuto rilevanti i seguenti interrogativi nello sviluppo del modello:

a) Il prezzo varia a seconda delle compagnie aeree?

b) Come viene influenzato il prezzo se i biglietti vengono acquistati solo 1 o 2 giorni prima della partenza?

c) Il prezzo del biglietto cambia in base all'orario di partenza e all'orario di arrivo?

d) Come cambia il prezzo al variare della Sorgente e della Destinazione?

e) Come varia il prezzo del biglietto tra Economy e Business Class?

Inoltre, la raccolta di dati attendibili è un ulteriore scoglio. Per questa task ho deciso di affidarmi ad operazioni di Web Scraping e manipolazione di set già esistenti, adattandoli ai requisiti del mio sistema.

Modellazione

Ho considerato due ipotesi per lo sviluppo del modello.

La prima, prevedeva di considerare la feature "price" come variabile target, e di conseguenza di astrarre il problema ad un problema di regressione.

In seguito, ho considerato di modellare il problema come una classificazione binaria ed considerare la feature "price" come variabile dipendente: il prezzo del biglietto scenderà? YES or NOT.

Una terza opzione che ho vagliato, è quella che prevede di combinare i due approcci: utilizzare la regressione per prevedere il prezzo futuro, ma mantenendo le caratteristiche di un problema di classificazione. In poche parole:

boolean price_decrease=1 se price_future<price_current, else =0.

Ho deciso di non modellare il problema come una classificazione binaria per la previsione del price_decrease di un biglietto aereo, poiché nel set di dati avevo a disposizione solo una feature legata al prezzo e non la sua progressione storica.

Modellazione scelta:

Un modello regressivo, in grado di stimare il costo di un biglietto aereo basandosi su determinate features del contesto. Il modello dovrà produrre un output all'utente che consigli se aspettare o meno per l'acquisto di un biglietto aereo. Alla prima release, il modello produrrà stime per una data di 7 giorni successiva a quella di utilizzo.

About the dataset

Ho effettuato uno scraping iniziale di applicazioni come SkyScanner utilizzando Selenium. Tuttavia, ho ritenuto i dati raccolti poco affidabili. Ho quindi deciso di affidarmi a un set di dati iniziale di Kaggle, "FlightFare", anch'esso ottenuto tramite web scraping:

"Lo strumento di scraping Octoparse è stato utilizzato per estrarre i dati dal sito web. I dati sono stati raccolti in due parti: una per i biglietti di classe economica e un'altra per i biglietti di classe business.

Dal sito è stato estratto un totale di 300261 distinte opzioni di prenotazione di voli per i viaggi aerei tra le 6 principali città metropolitane dell'India.

I dati sono stati raccolti per 50 giorni, dall'11 febbraio al 31 marzo 2022.

La fonte dei dati erano dati secondari e sono stati raccolti dal sito web Ease my trip".

Limiti del dataset

- **Timeline**: La mancanza di dati cronologici continui (es. fluttuazioni giornaliere dettagliate) limita la capacità del modello di cogliere pattern temporali più complessi.
- **Bias geografico**: La concentrazione di dati su sei città indiane potrebbe rendere il modello meno generalizzabile ad altre regioni.

FEATURES

Il dataset si presenta con 12 features, illustrate di seguito:

0) **Unnamed 0:** feature non ben specificata. Probabilmente indica l'id del volo. Questa feature verrà scartata perché considerata non rilevante.

1) **airline**: il nome della compagnia aerea, caratteristica categorica con 6 possibili valori all'interno del set.

2) **Flight**: Memorizza le informazioni relative al codice di volo dell'aereo, feature categorica.

3) **Source city**: Città da cui decolla il volo, feature categorica con 6 città uniche.

4) **Departure time**: si tratta di una caratteristica categorica derivata, ottenuta attraverso la discretizzazione dei valori in fasce temporali significative. Memorizza le informazioni sull'orario di partenza e dispone di 6 etichette orarie univoche.

5) **Stops**: una funzione categoriale con valori distinti che memorizza il numero di scali tra le città di origine e di destinazione.

6) **Arrival time**: si tratta di una caratteristica categorica derivata, ottenuta attraverso la discretizzazione dei valori in fasce temporali significative. Memorizza le informazioni sull'orario di arrivo e dispone di 6 etichette orarie univoche.

7) **Destination city**: Città dove atterrerà il volo. È una caratteristica categorica con 6 città uniche.

8) **Class**: una caratteristica categorica che contiene informazioni sulla classe di viaggio; ha due valori distinti: Business ed Economy.

9) **Duration**: una valore continuo che mostra il tempo complessivo necessario per viaggiare tra le città in ore.

10) **Days_left**: questa è una feature derivata che viene calcolata sottraendo la data del viaggio dalla data di prenotazione.

11) **Price**: memorizza le informazioni sul prezzo del biglietto.

Starter set

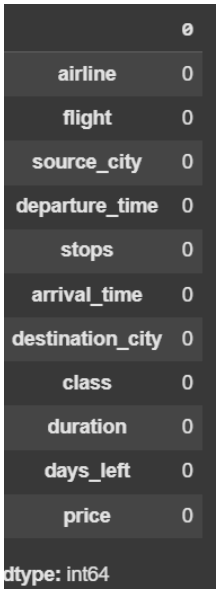
	Unnamed: 0	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953
1	1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953
2	2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956
3	3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955
4	4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955

Feature engineering

La seguente sezione illustra la seconda fase della pipeline di sviluppo, ossia quella di feature engineering. All'interno del processo, ho esplorato il set di dati a disposizione al fine di valutare se e come manipolare le feature a disposizione per facilitare il lavoro del modello. Ho, quindi, cercato eventuali valori mancanti, analizzato possibili correlazioni e pattern nascosti, gestito le feature numeriche e categoriche.

Valori mancanti

Analizzando il set di dati, non risulta necessaria nessuna tecnica di imputazione: potendo contare su un dataset molto nutrito, i pochi record (3) che presentavano un valore mancante sono stati eliminati, senza comportare distorsioni nella rappresentazione dei dati.



	0
airline	0
flight	0
source_city	0
departure_time	0
stops	0
arrival_time	0
destination_city	0
class	0
duration	0
days_left	0
price	0

dtype: int64

Analisi statistica delle feature numeriche

	duration	days_left	price
count	300153.000000	300153.000000	300153.000000
mean	12.221021	26.004751	20889.660523
std	7.191997	13.561004	22697.767366
min	0.830000	1.000000	1105.000000
25%	6.830000	15.000000	4783.000000
50%	11.250000	26.000000	7425.000000
75%	16.170000	38.000000	42521.000000
max	49.830000	49.000000	123071.000000

Tipi di dati

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   airline           300153 non-null  object
 1   flight            300153 non-null  object
 2   source_city       300153 non-null  object
 3   departure_time    300153 non-null  object
 4   stops             300153 non-null  object
 5   arrival_time      300153 non-null  object
 6   destination_city  300153 non-null  object
 7   class             300153 non-null  object
 8   duration          300153 non-null  float64
 9   days_left         300153 non-null  int64
 10  price             300153 non-null  int64
dtypes: float64(1), int64(2), object(8)
memory usage: 25.2+ MB
```

Analisi di correlazioni e pattern

Nella sezione seguente, riporto i grafici ottenuti manipolando il set di dati, al fine di fornire una rappresentazione visuale dell'indagine condotta rispetto alle possibili correlazioni tra le varie feature e la loro distribuzione. I risultati evidenziati dalle rappresentazioni visuali sono stati la base per le scelte di progettazione.

Grafico 1: numero di voli rispetto alla Compagnia aerea all'interno del dataset.

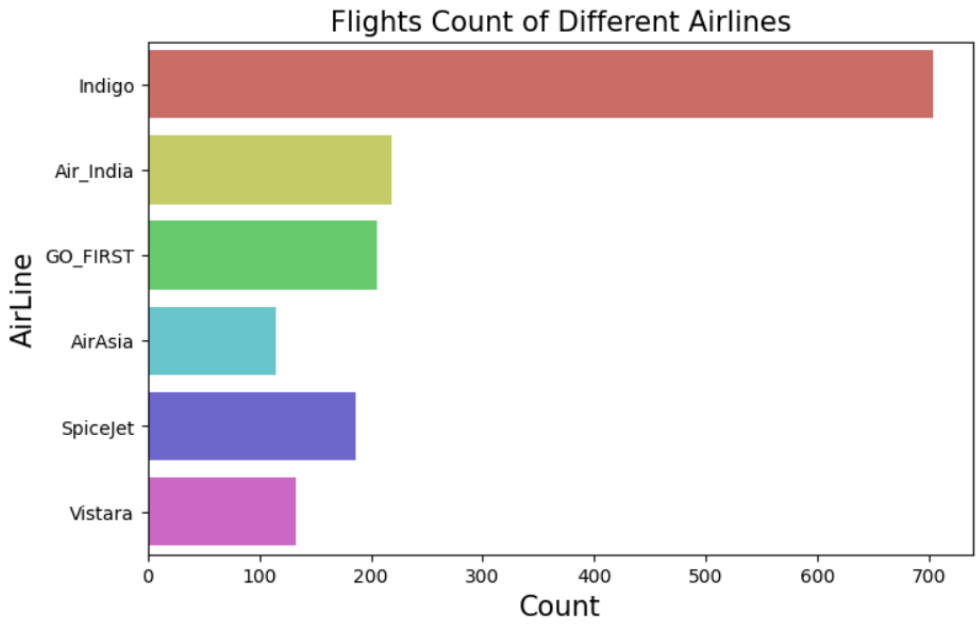


Grafico 2: percentuale di voli rispetto alla classe di viaggio all'interno del

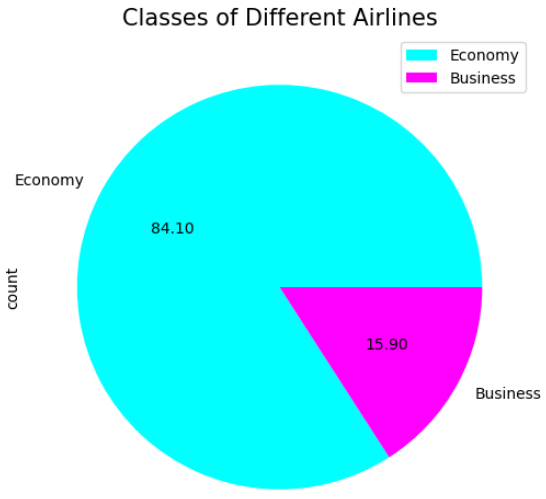


Grafico 3: costo del biglietto rispetto alla compagnia aerea.

- Come varia il prezzo a seconda della compagnia aerea? E' saggio discretizzare la feature in una caratteristica binomiale (Low-cost, Standard)?

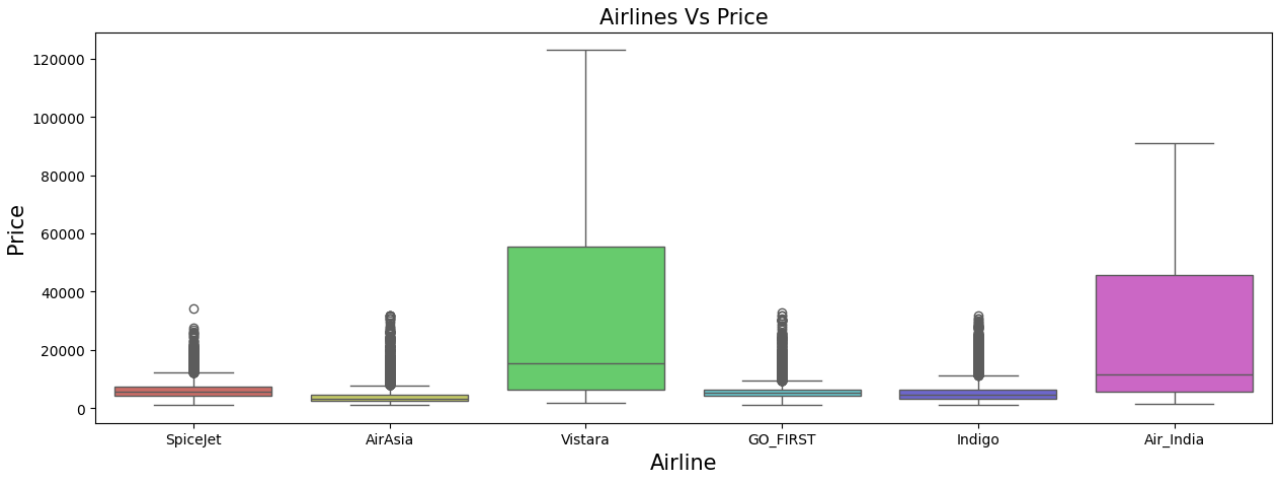


Grafico 4: distribuzione del prezzo rispetto al tipo di biglietto.

- Come varia il prezzo a seconda della classe di viaggio?
  - Come prevedibile, il prezzo per un biglietto business è sensibilmente maggiore rispetto a un biglietto in Economy.

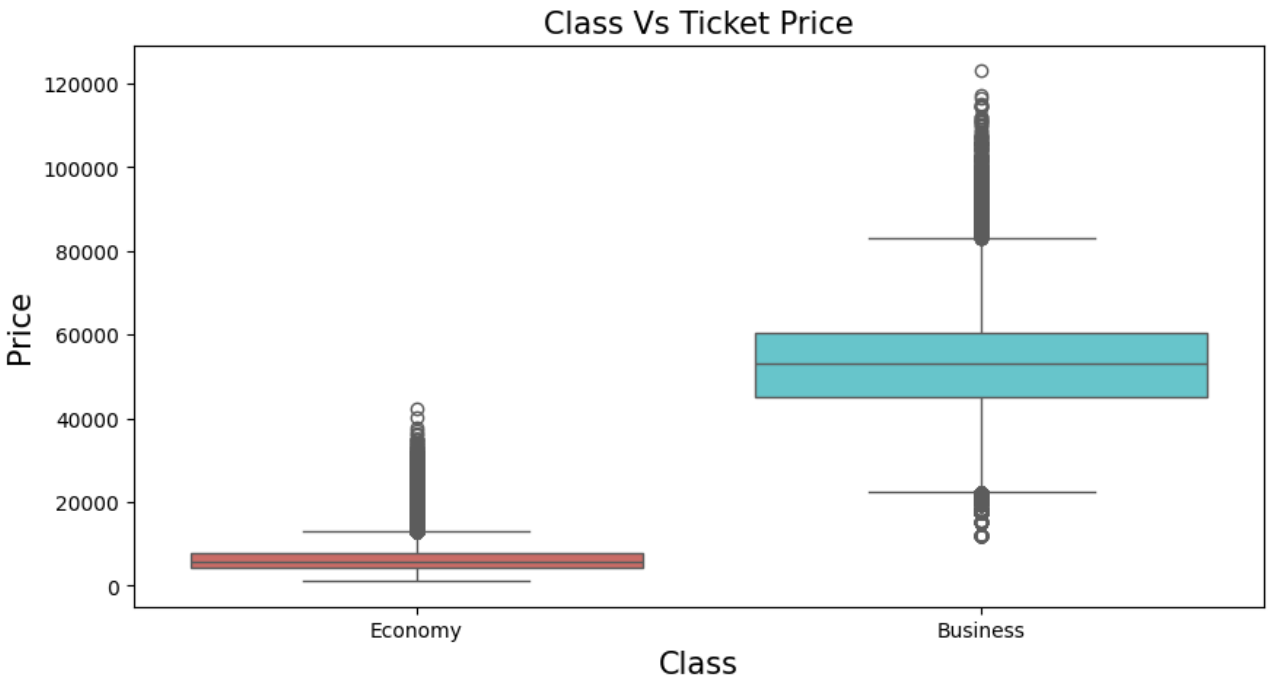


Grafico 5: distribuzione di price rispetto al numero di scali del volo.

- ❖ Che correlazione c'è tra il prezzo del biglietto e il numero di scali previsti?
  - I voli con uno scalo hanno il prezzo massimo maggiore.

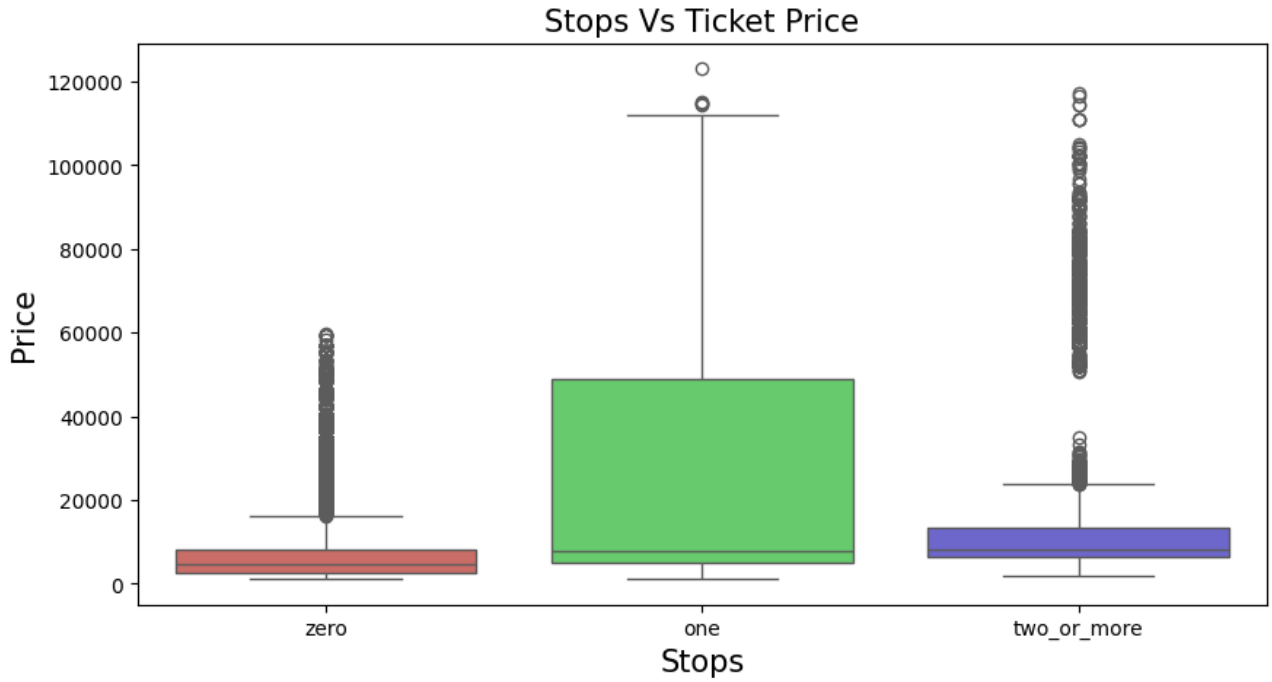
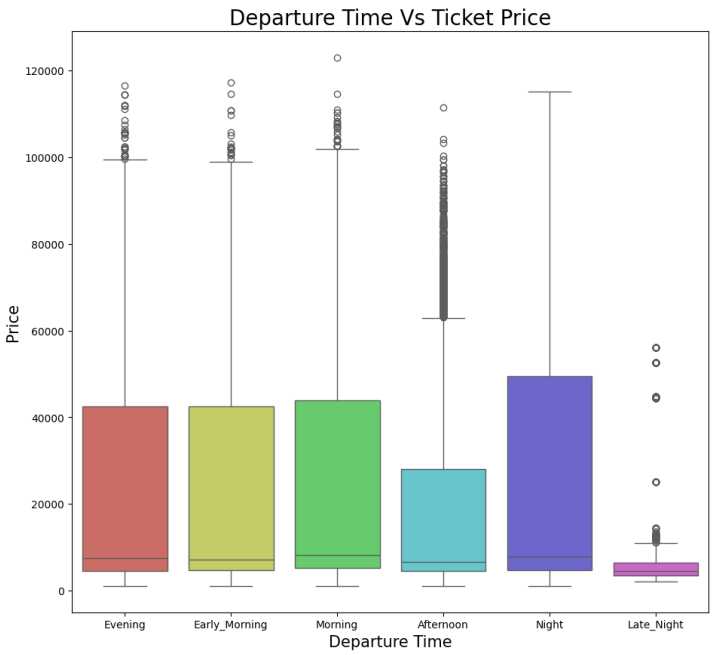
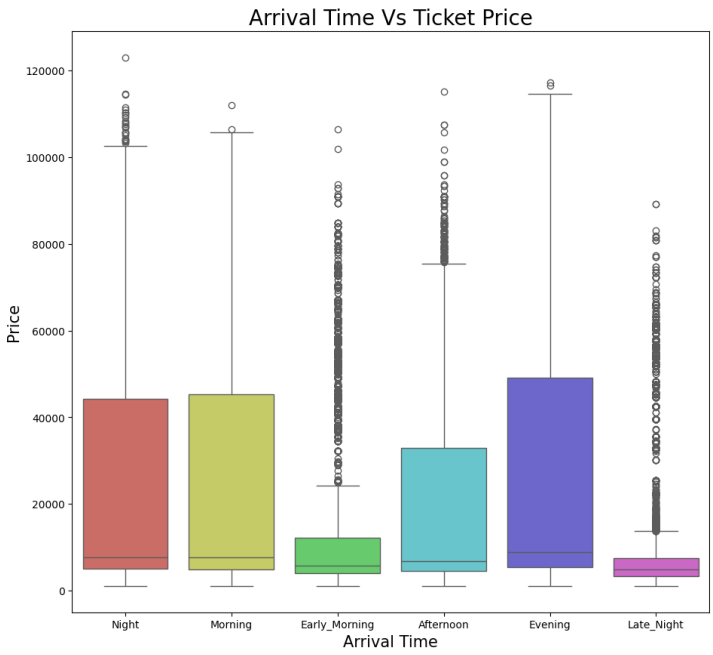


Grafico 6: distribuzione di price rispetto all'orario di partenza e all'orario di arrivo previsti.

❖ L’orario di partenza e quello di arrivo influenzano il prezzo?





1. Orario di partenza e prezzo del biglietto
    - Il prezzo del biglietto è maggiore per i voli quando l'orario di partenza è notturno.
    - Il prezzo del biglietto è quasi uguale per i voli con orario di partenza al mattino presto, al mattino e alla sera.
    - Il prezzo del biglietto è basso per i voli con orario di partenza in tarda notte.
2. Orario di arrivo e prezzo del biglietto
    - Il prezzo del biglietto è maggiore per i voli quando l'orario di arrivo è serale.
    - Il prezzo del biglietto è quasi uguale per i voli con orario di arrivo al mattino e alla sera.
    - Il prezzo del biglietto è basso per i voli con orario di arrivo a tarda notte uguale all'orario di partenza.

Grafico 7: Distribuzione di price rispetto alla rotta.

❖ Analisi dei luoghi di partenza e di arrivo
- Kolkata è la città più cara, sia come punto di partenza che di arrivo.
- Mumbai, Chennai , Hyderabad e Bangalore hanno range di prezzo simili.
- Delhi è la città più economica, sia per la partenza che per la destinazione.

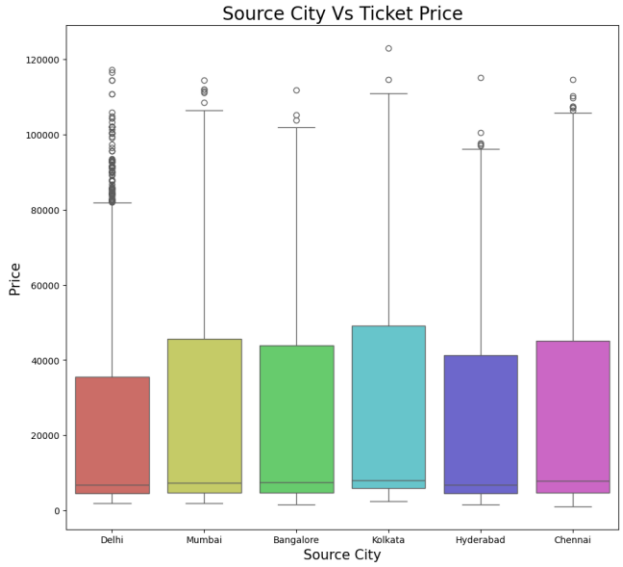


Grafico 8: Price rispetto al variare della durata del volo.

❖ La durata del volo come influisce sul prezzo?
- All'aumentare della durata il prezzo del biglietto sale sia per i voli Economy che per quelli Business.

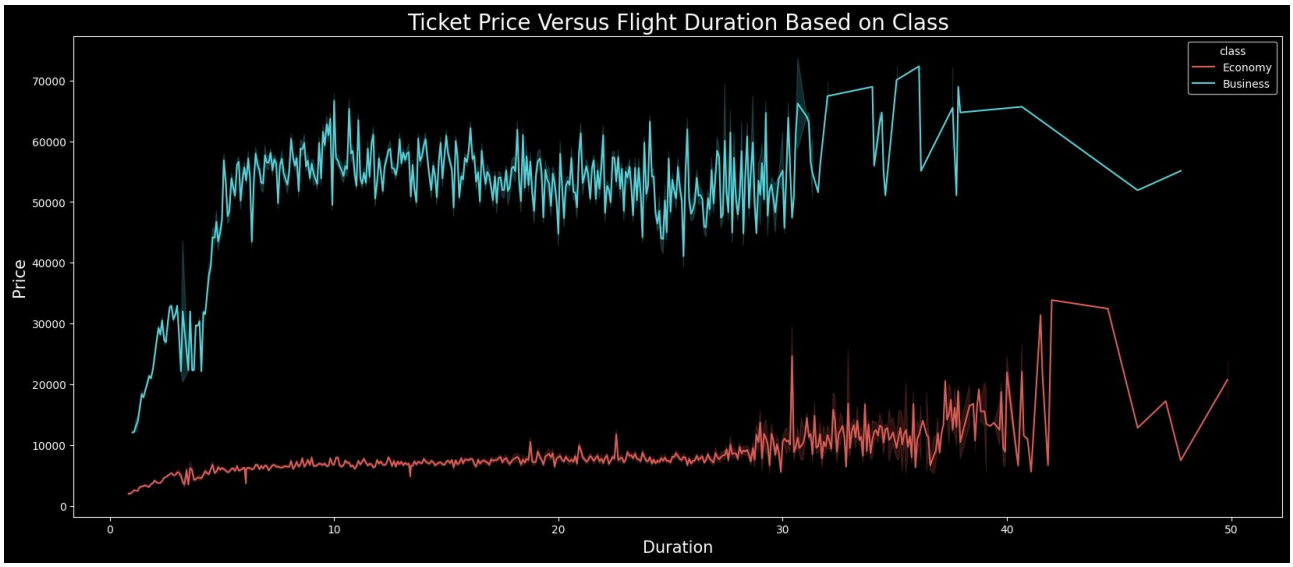
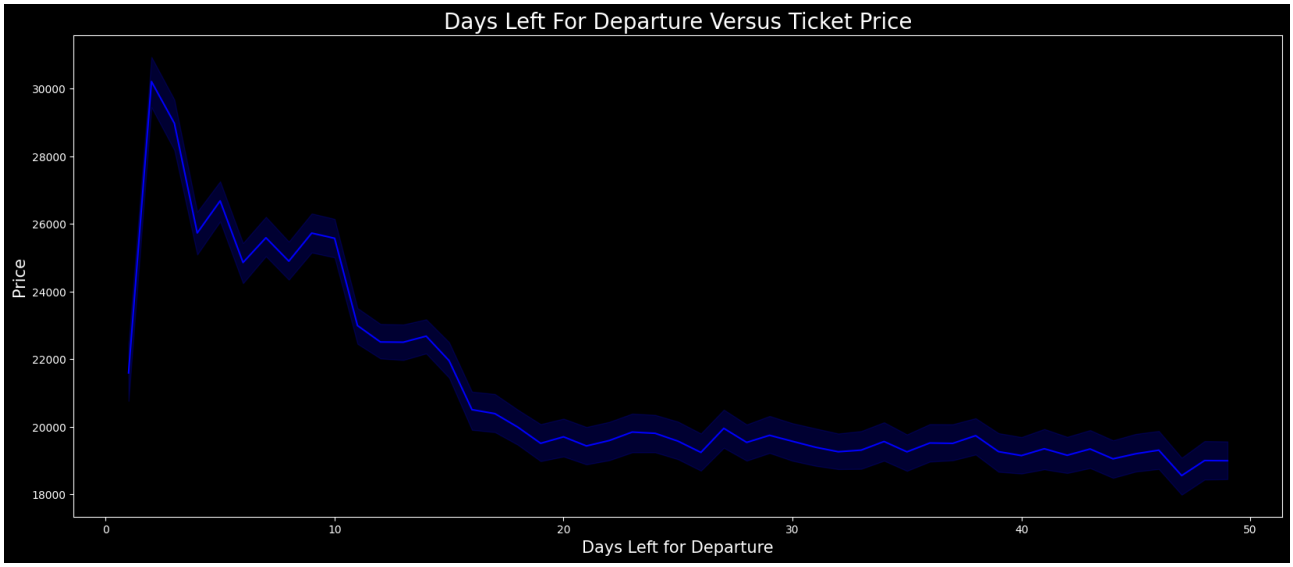


Grafico 9: Rapporto tra anticipo della prenotazione e prezzo.

❖ L'anticipo con cui si intende prenotare come influisce sul prezzo?

- Il prezzo è massimo a pochi giorni dalla partenza.



Gestione degli Outlier

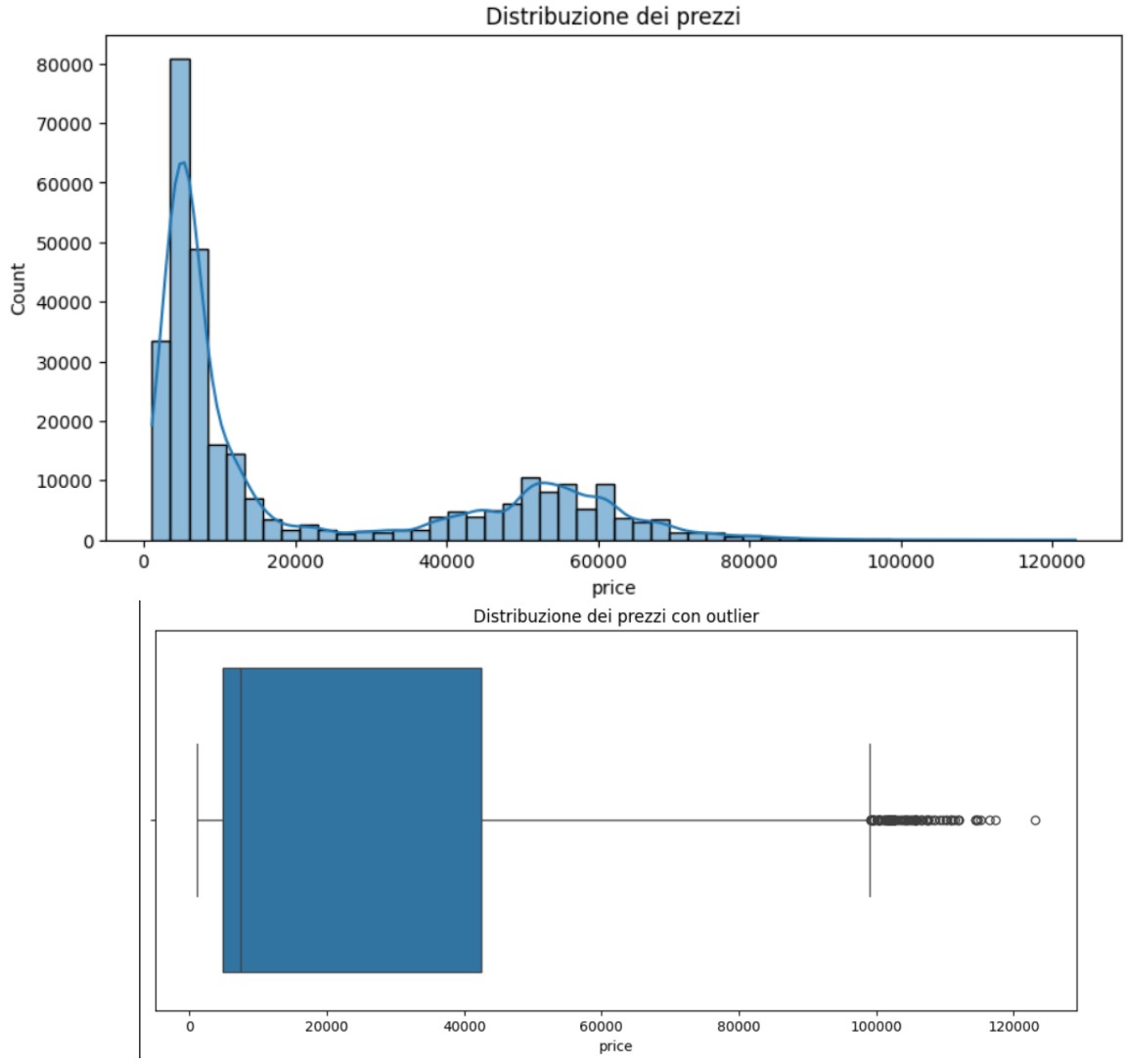
Durante l'analisi esplorativa dei dati, è stata osservata una presenza limitata di outlier nei prezzi dei biglietti aerei. Per garantire la qualità dei dati e migliorare le prestazioni del modello di machine learning, si è deciso di adottare una strategia di pulizia basata su criteri statistici.

Analisi degli outlier

L'analisi degli outlier è stata condotta utilizzando due metodi:

- Interquartile Range (IQR): ha identificato 123 outlier (0,04% del dataset).
- Z-score: ha identificato 602 outlier (0,20% del dataset).
- Poiché il numero di valori anomali è molto ridotto, essi non influenzano significativamente il dataset. Tuttavia, la loro presenza potrebbe distorcere leggermente il modello, quindi ho optato per una strategia di pulizia.

Grafico 10, 11: Distribuzione dei prezzi.



Rimozione degli outlier

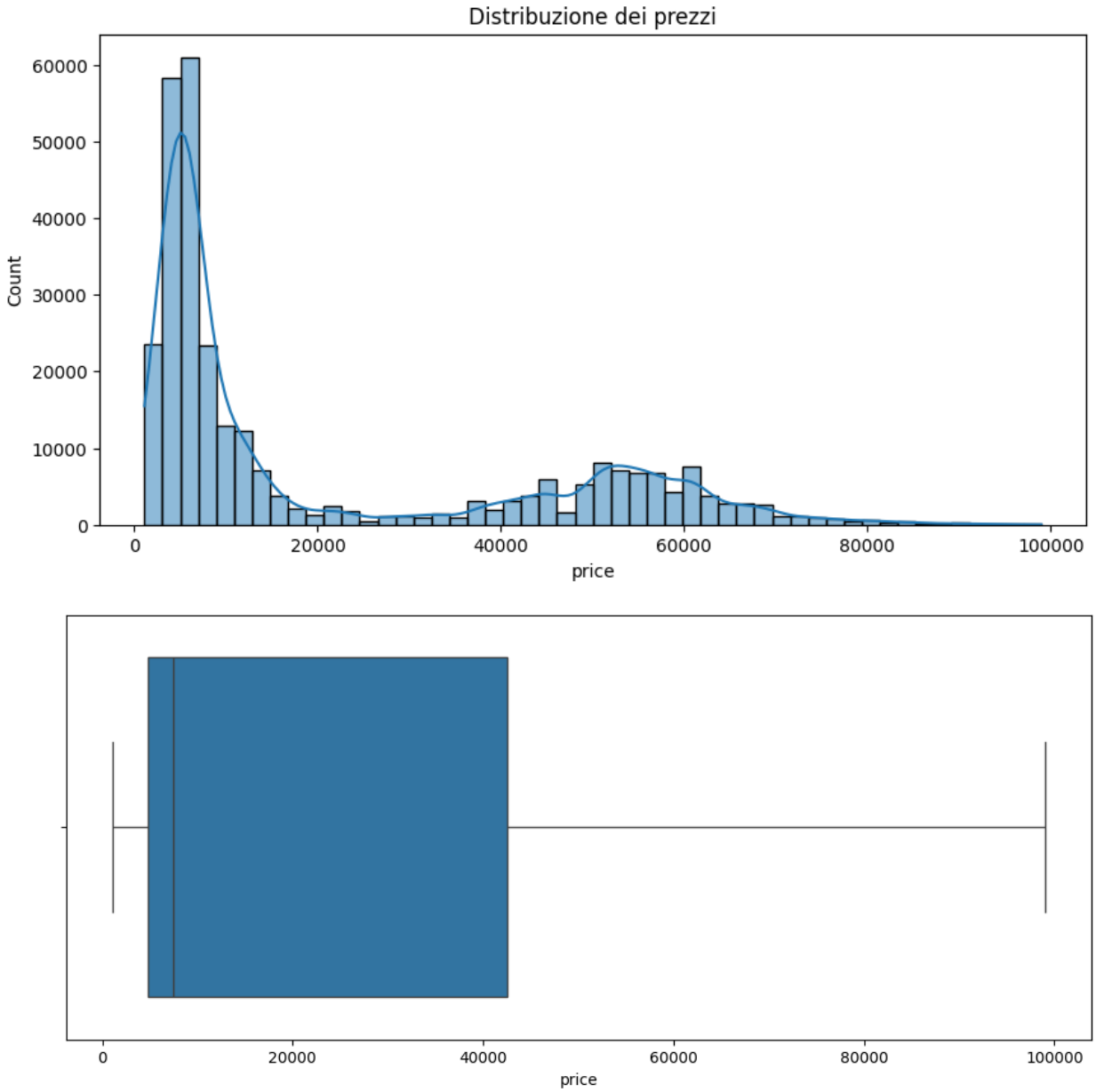
Dopo aver confrontato diverse opzioni, si è deciso di rimuovere gli outlier identificati con il metodo IQR, poiché essi sono una percentuale molto bassa del dataset (<1%), dunque non si perde una quantità significativa di dati utili. Inoltre, si garantisce una distribuzione più stabile della variabile (price).

La pulizia è stata effettuata escludendo i prezzi al di fuori dell'intervallo interquartile (Q1 - 1.5 * IQR, Q3 + 1.5 * IQR). Il dataset pulito è stato poi salvato in un nuovo file per essere utilizzato nei passaggi successivi del progetto.

Dimensione originale: 300153 righe

Dimensione dopo la rimozione degli outlier: 300030 righe

Grafico 12, 13: distribuzione dei prezzi dopo la rimozione degli outlier.

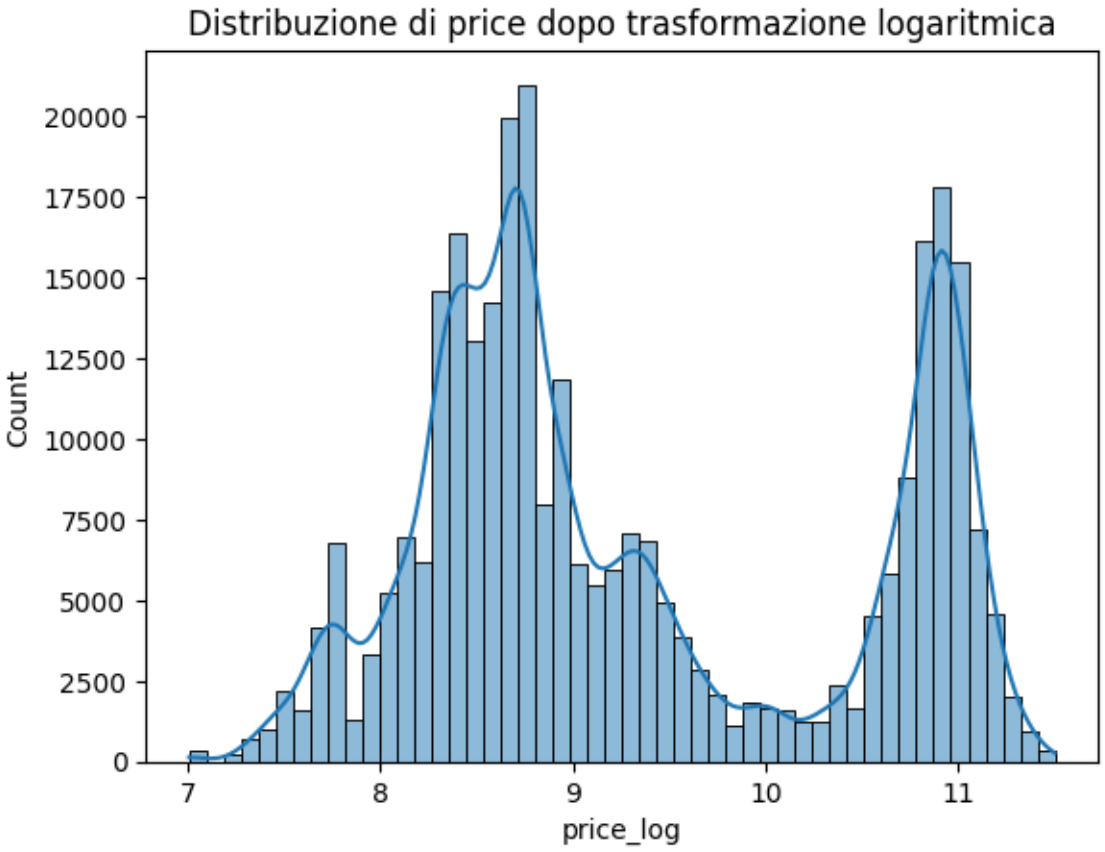


Trasformazione logaritmica su price

Nonostante la rimozione degli outlier, la distribuzione di price presenta ancora un’evidente asimmetria con coda lunga a destra. Per migliorare la stabilità del modello e ridurre l’impatto degli outlier, ho deciso di applicare una trasformazione logaritmica, al fine di:

- Ridurre la skewness.
- Migliore rappresentazione della relazione tra price e altre variabili.
- Adattare i dati ai requisiti di modelli sensibili alla scala (come i modelli regressivi).

Grafico 14: distribuzione di price dopo la trasformazione logaritmica.



Discretizzazione

Studi sui prezzi dei voli aerei mostrano una variazione significativa dei prezzi in base ai giorni mancanti alla partenza. Per rappresentare meglio (e in modo più semplice per il modello) questa relazione, ho deciso di effettuare una discretizzazione in fasce orarie della feature “days_left”, che rappresenta il numero di giorni che mancano alla partenza del volo.

Non ho utilizzato né la discretizzazione a ugual ampiezza, né quella ad ugual frequenza: ho invece deciso di definire degli intervalli discreti di valori continui basandomi su osservazioni statistiche di trend di piattaforme come “Skyscanner”.

- **Intervalli individuati**:
    - **0-7 giorni**: Finestra last-minute con prezzi più alti.
    - **8-21 giorni**: Calo progressivo dei prezzi.
    - **22-50 giorni**: Finestra ottimale per prezzi bassi.
    - **51-100 giorni**: Prenotazioni in anticipo, prezzi stabili.

Come si può notare dalla tabella in figura, il set di dati non è ben nutrito rispetto a soluzioni di volo da prenotare nell’intervallo 51-100 (days_left). Per sviluppi futuri si considera la possibilità di effettuare data augmentation al fine di colmare questa lacuna.

Tuttavia, confrontando le prestazioni del modello con la feature “days_left” rispetto a prestazioni del modello con la feature “days_left_bin”, la prima configurazione si è dimostrata significativamente migliore. Per cui, ho deciso di lasciare invariata la feature originale.

days_left_bin	count
22-50	180074
8-21	87928
0-7	32028
51-100	0

La feature “airline”

Come si evince dal Grafico 3, il prezzo del biglietto è fortemente influenzato dalla compagnia aerea che offre il servizio. All’interno dei dati analizzati, “Vistara” e “AirIndia”

risultano essere le compagnie più costose. “SpiceJet”, “AirAsia”, “GO_FIRST” ed “Indigo”, invece, hanno una distribuzione dei prezzi pressappoco simile.

Per questo motivo, ho deciso di ridurre il numero di valori possibili della feature “Airline” a due: “Standard” (Visitara e AirIndia) e “Low – Cost” (le altre).

I test hanno evidenziato risultati migliori nella configurazione con valori continui della feature.

Codifica delle feature categoriche

Nel contesto del progetto, le feature categoriche rappresentano variabili che contengono valori discreti e non numerici, come nomi di città, compagnie aeree, orari di partenza, ecc. Poiché la maggior parte degli algoritmi di machine learning richiede input numerici, è necessario trasformare queste feature categoriche in un formato numerico.

Ho scelto di utilizzare il Label Encoding per le seguenti ragioni:

Semplicità e Efficienza: Il Label Encoding è una tecnica semplice e diretta che assegna un numero intero unico a ciascuna categoria. Questo metodo è computazionalmente efficiente e facile da implementare.

Compatibilità con Algoritmi di Machine Learning: Molti algoritmi di machine learning, come Random Forest, possono gestire efficacemente le feature codificate con il Label Encoding, poiché non assumono un ordine tra le categorie.

Riduzione della Dimensionalità: A differenza del One-Hot Encoding, che può aumentare significativamente la dimensionalità del dataset, già molto nutrito nel mio caso (soprattutto quando ci sono molte categorie uniche), il Label Encoding mantiene il numero di feature invariato. Questo è particolarmente utile quando si lavora con dataset di grandi dimensioni o con molteplici feature categoriche.

Preservazione dell'Informazione: Il Label Encoding assegna un valore numerico a ciascuna categoria senza perdere informazioni sulle categorie stesse. Questo è importante per mantenere l'integrità dei dati durante il processo di trasformazione.

Normalizzazione

I dati relativi alle feature numeriche sono stati normalizzati utilizzando il MinMaxScaler, a seguito della divisione in training set e test set.

Addestramento

Successivamente, i dati sono stati divisi in **training set (70%) e test set (30%)** e normalizzati utilizzando il **MinMaxScaler**.

2. Modelli Utilizzati e Risultati

Sono stati testati diversi modelli di regressione per prevedere il prezzo futuro dei biglietti. La tabella seguente riassume le principali metriche ottenute:

Modello	MAE	RMSE	R^2	MAPE
Linear Regression	4588	6980	0.904	43.1%

Modello	MAE	RMSE	R^2	MAPE
Decision Tree	1449	2997	0.982	13.5%
Random Forest	1437	2889	**0.9836**	13.5%
K-Nearest Neighbors	1585	3181	0.980	14.5%
Extra Trees	1440	2958	0.982	13.4%

MAE

Il MAE è molto basso per **Random Forest**, indicando un errore medio di circa **1437** sul prezzo predetto. Dato che i prezzi dei biglietti possono variare di migliaia di unità, questo è un buon valore.

RMSE

🔹 **Risultato**: Il valore **2889** per il modello Random Forest è buono, ma significa che in media il modello può sbagliare il prezzo del biglietto di circa **2889 unità monetarie**.

R^2

Anche per questa metrica, Random Forest risulta la scelta migliore, con un valore di **0.9836**. Significa che il modello è in grado di spiegare circa **98.36% della variazione del prezzo**.

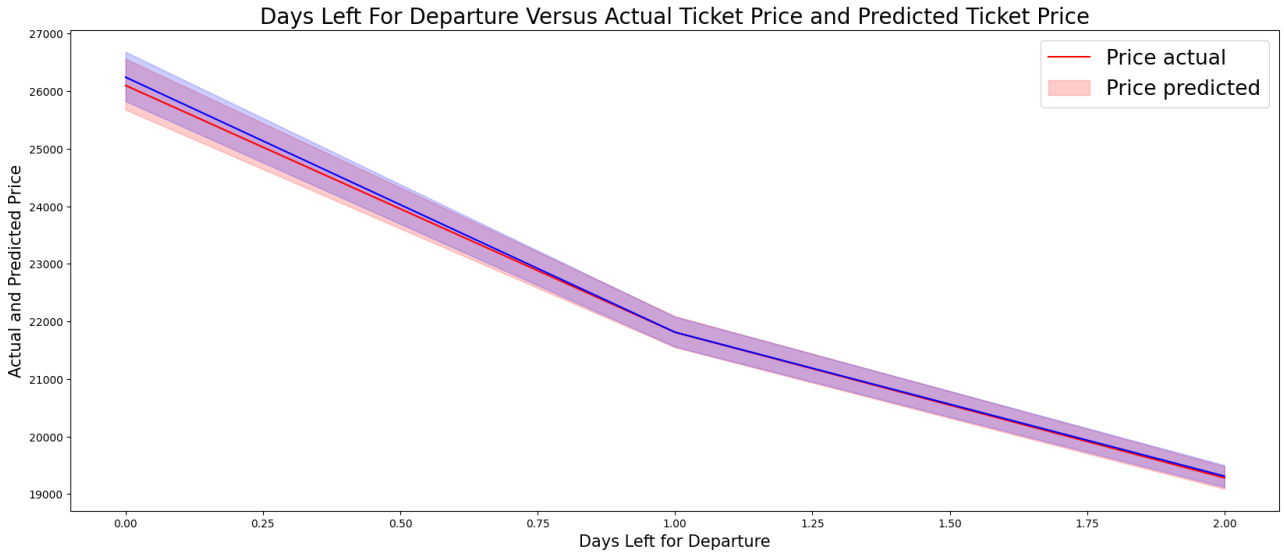
MAPE

🔹 **Risultato**: Il modello Random Forest commette **in media un errore del 13.5%** sul prezzo dei biglietti. Questo è un risultato **accettabile**, ma migliorabile.

Il **Random Forest Regressor** è risultato il modello migliore, con **MAE = 1437** e **R^2 = 0.9836**, indicando un'elevata capacità predittiva.

Confronto tra Prezzi Reali e Predetti

Per verificare le performance del modello, è stato realizzato un grafico che confronta i prezzi reali e quelli previsti in base ai giorni mancanti alla partenza. Il modello è in grado di catturare correttamente l'andamento dei prezzi nel tempo.



HyperParameters Tuning: Grid Search

L'obiettivo di questa fase del progetto è stato quello di ottimizzare il modello di regressione Random Forest, migliorandone le prestazioni attraverso la ricerca dei migliori iperparametri utilizzando la tecnica della **Grid Search con Cross-Validation**. Questo approccio mi ha permesso di esplorare in modo sistematico combinazioni di iperparametri, selezionando quella che massimizza le prestazioni del modello su dati di test non visti.

Ho scelto la **Grid Search** perché attraverso l' **esplorazione sistematica** permette di testare tutte le combinazioni di valori iperparametrici predefiniti. Inoltre, Grid Search mi ha permesso di garantire maggior robustezza, in combinazione con la Cross-Validation, riducendo il rischio di overfitting o underfittin e garantendo un modello più generalizzabile. Inoltre, essa consente una valutazione rigorosa e replicabile delle configurazioni.

La Random Forest presenta numerosi iperparametri che influenzano le sue prestazioni, tra cui il numero di alberi, la profondità massima e i criteri per suddividere i nodi. La Grid Search è stata utilizzata per trovare una combinazione ottimale che bilanciasse complessità e accuratezza.

Ho quindi definito uno spazio di ricerca iperparametrico come segue:

- **max_depth**: 10,20,30,None
- **min_samples_split**: 2,5,10
- **min_samples_leaf**: 1,2,4
- **n_estimators**: 50,100,200

La ricerca ha valutato tutte le combinazioni possibili (72 in totale), utilizzando una **cross-validation a 5 fold** per ogni configurazione. Ciò ha garantito che ogni combinazione venisse valutata su diversi sottoinsiemi del training set, riducendo il rischio di sovradattamento ai dati di addestramento.

Dopo la valutazione, la combinazione ottimale di iperparametri selezionata è stata:

- **max_depth**: 20
- **min_samples_split**: 5
- **min_samples_leaf**: 1
- **n_estimators**: 100

Questa configurazione ha dimostrato di fornire un buon equilibrio tra **bias** e **varianza**, massimizzando le prestazioni del modello senza aumentare eccessivamente la complessità computazionale. L'applicazione di questi parametri ha permesso di ottenere le seguenti valutazioni:

Metrica	Mod. Iniziale	Mod. Ottimizzato
MAE	1437	**1194.26**
RMSE	2889	**2,582.56**
R^2 Score	0.9836	**0.9869**

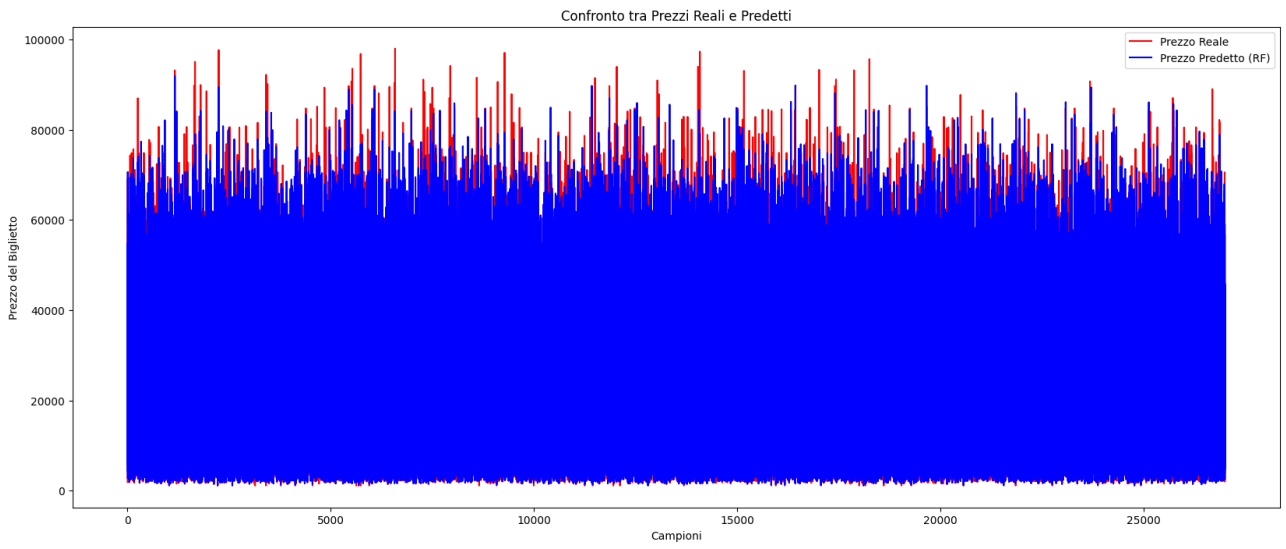
- **MAE** (Mean Absolute Error): La riduzione dell'errore medio assoluto indica che il modello ottimizzato produce previsioni più accurate.
- **MSE** e **RMSE**: Un miglioramento significativo dimostra che le predizioni sono più vicine ai valori reali anche nei casi con errori più grandi.
- **R² Score**: Il punteggio di 0.9869 mostra che il modello spiega quasi tutta la variabilità dei dati, indicando buone capacità predittive.

Interpretazione dell'Ottimizzazione

L'ottimizzazione ha migliorato la capacità del modello di generalizzare a nuovi dati.

- L'aumento della profondità massima (**max_depth**) ha permesso agli alberi di catturare meglio pattern complessi, evitando però overfitting grazie alla restrizione su **min_samples_split** e **min_samples_leaf**.
- Un numero maggiore di stimatori (**n_estimators**) ha migliorato la robustezza del modello, riducendo la varianza nelle previsioni.

Il grafico sottostante mostra un confronto tra i **prezzi reali** (linea rossa) e quelli **predetti dal modello ottimizzato** (linea blu). È evidente come il modello ottimizzato segua con precisione l'andamento reale, con discrepanze minime, confermando la validità dell'approccio.



Deployment

Ho utilizzato Streamlit per deployare l’applicazione in locale. Il poco tempo a disposizione per curare il deploy ha comportato un problema rispetto al numero di feature con cui il modello lavora: in fase di addestramento lavora con 10 features, in fase di deploy invece con 9.

Visti i tempi stretti, non sono riuscito a correggere il bug entro l’ora di consegna.