UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

DIPARTIMENTO DI INFORMATICA

CASO DI STUDIO

INGEGNERIA DELLA CONOSCENZA

AA 23/24

A Machine Learning Approach To AirBnB

Autore:

Nicola Visci

Matricola: 700797

E-mail: n.visci2@studenti.uniba.it

Docente:

Prof. Nicola Fanizzi

GitHub Repository

https://github.com/NicoVisci/ProgettoIcon/tree/master

Introduzione

Negli ultimi anni, il mercato degli affitti a breve termine ha visto una crescita esponenziale, trainata in gran parte dall'ascesa di piattaforme come Airbnb. Queste piattaforme offrono ai proprietari la possibilità di affittare case, appartamenti e stanze a turisti e viaggiatori, generando un mercato dinamico e competitivo. Tuttavia, determinare il prezzo ideale per un annuncio su Airbnb è una sfida complessa, influenzata da una vasta gamma di fattori, tra cui la posizione, le caratteristiche dell'alloggio, la stagionalità, e le recensioni degli ospiti.

In questo contesto, la previsione accurata del prezzo di affitto è fondamentale sia per i proprietari che per i potenziali ospiti. Da un lato, i proprietari desiderano massimizzare i loro profitti senza scoraggiare i potenziali clienti con prezzi troppo elevati. Dall'altro, i viaggiatori cercano il miglior rapporto qualità-prezzo, bilanciando le loro esigenze con il budget disponibile.

Il progetto che segue utilizza tecniche di machine learning per affrontare questa sfida. L'obiettivo principale è sviluppare un modello predittivo che possa stimare accuratamente il prezzo di affitto di un annuncio su Airbnb, basandosi su un insieme di caratteristiche rilevanti. Analizzando un dataset di annunci Airbnb, il progetto esplora vari algoritmi di machine learning, tra cui Random Forest, per identificare i fattori chiave che influenzano i prezzi e migliorare la precisione delle previsioni.

Per questo studio, ci si è posti l'obbiettivo di fornire previsioni accurate di prezzo di affitto di annunci Airbnb in una determinata città, non solo fornendo ai potenziali clienti informazioni utili per confrontare le opzioni disponibili, in base alle proprie esigenze, ma anche strumenti che possano guidare i proprietari in una consapevole definizione delle loro strategie di pricing, aumentando la loro competitività e contribuendo a rendere il mercato degli affitti a breve termine più trasparente ed efficiente.

Per lo sviluppo software del progetto si è utilizzato il linguaggio Python, particolarmente vantaggioso grazie alla ricchezza di librerie e strumenti disponibili, che semplificano il processo di sviluppo, analisi e modellazione.

Innanzitutto, si è usufruito di **Pandas**, libreria fondamentale per la manipolazione e l'analisi dei dati. Con Pandas, possiamo facilmente caricare dataset di grandi dimensioni, eseguire operazioni di pulizia dei dati, trasformare variabili, ed esplorarli. La sua struttura del DataFrame permette di gestire i dati in modo simile a un foglio di calcolo, rendendo il processo di analisi molto intuitivo anche per chi ha poca esperienza.

Quando si tratta di costruire e valutare modelli di machine learning, **scikit-learn (sklearn)** è uno degli strumenti più potenti e flessibili disponibili in Python. Questa libreria offre una vasta gamma di algoritmi di machine learning, dalle regressioni alle foreste casuali e alle reti neurali, con un'interfaccia coerente e facile da usare. Sklearn semplifica anche l'implementazione di tecniche di pre-processing, come la standardizzazione dei dati e l'encoding delle variabili categoriche, essenziali per preparare i dati prima dell'allenamento del modello.

Inoltre, sklearn include strumenti per la valutazione dei modelli, come la cross-validation, e metriche di performance come l'errore quadratico medio (MSE) e il coefficiente di determinazione (R^2), che ci permettono di quantificare l'accuratezza del modello e ottimizzare i parametri per migliorare le previsioni.

In sintesi, la combinazione di Pandas per la gestione dei dati e sklearn per la modellazione e la valutazione, rende Python un linguaggio ideale per il nostro caso d’uso

1. Dataset

Il dataset di riferimento per questo progetto è stato scaricato dal repository di open data di Kaggle: contiene dati relativi alle inserzioni reali nell'area di New York City estratte dal sito di Airbnb. Il dataset include, tra le altre, diverse caratteristiche rilevanti per ciascun annuncio, come la posizione, il tipo di camera, il numero e la qualità di recensioni, la disponibilità ed il prezzo.

Il dataset è stato creato con l'intenzione di supportare le attività di analisi a tutti i livelli, per questo è composto da un numero bilanciato di entries rispetto ai dati che mette in luce, ottimizzando la sua adattabilità al contesto del machine learning; difatti ha ottenuto un punteggio di usabilità eccezionale su Keggle, pari a 10.

Il dataset è disponibile al seguente link:

https://www.kaggle.com/datasets/arianazmoudeh/airbnbopendata/data

I dati necessitavano comunque di essere maneggiati e formattati prima di qualsiasi utilizzo poiché pensati anche per supportare il data cleaning. Perciò è stata condotta un'attenta analisi sulla struttura e la natura dei dati presenti e sono stati effettuate operazioni ad hoc per gestire le incongruenze, rimuovere informazioni ridondanti e sostituire i valori mancanti o mal formattati.

1.1 Caricamento e Esplorazione dei Dati

I dati sono stati caricati in un DataFrame utilizzando la libreria Pandas. Il dataset originale è in formato CSV.

Il dataset è strutturato in 26 variabili di vario tipo come in figura:

```
RangeIndex: 102599 entries, 0 to 102598
Data columns (total 26 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   id                              102599 non-null  int64
 1   NAME                            102349 non-null  object
 2   host id                         102599 non-null  int64
 3   host_identity_verified          102310 non-null  object
 4   host name                       102193 non-null  object
 5   neighbourhood group             102570 non-null  object
 6   neighbourhood                   102583 non-null  object
 7   lat                             102591 non-null  float64
 8   long                            102591 non-null  float64
 9   country                         102067 non-null  object
 10  country code                    102468 non-null  object
 11  instant_bookable                102494 non-null  object
 12  cancellation_policy             102523 non-null  object
 13  room type                       102599 non-null  object
 14  Construction year               102385 non-null  float64
 15  price                           102352 non-null  object
 16  service fee                     102326 non-null  object
 17  minimum nights                  102190 non-null  float64
 18  number of reviews               102416 non-null  float64
 19  last review                     86706 non-null   object
 20  reviews per month               86720 non-null   float64
 21  review rate number              102273 non-null  float64
 22  calculated host listings count  102280 non-null  float64
 23  availability 365                102151 non-null  float64
 24  house_rules                     50468 non-null   object
 25  license                         2 non-null       object
dtypes: float64(9), int64(2), object(15)
```

Successivamente è stata condotta un'analisi statistica per comprendere meglio le variabili, presenti sia in formato numerico che categorico, e la loro distribuzione.

	id	host id	lat	long	Construction year	minimum nights	number of reviews	reviews per month	review rate number	calculated host listings count	availability 365
count	1.025990e+05	1.025990e+05	102591.000000	102591.000000	102385.000000	102190.000000	102416.000000	86720.000000	102273.000000	102280.000000	102151.000000
mean	2.914623e+07	4.925411e+10	40.728094	-73.949644	2012.487464	8.135845	27.483743	1.374022	3.279106	7.936605	141.133254
std	1.625751e+07	2.853900e+10	0.055857	0.049521	5.765556	30.553781	49.508954	1.746621	1.284657	32.218780	135.435024
min	1.001254e+06	1.236005e+08	40.499790	-74.249840	2003.000000	-1223.000000	0.000000	0.010000	1.000000	1.000000	-10.000000
25%	1.508581e+07	2.458333e+10	40.688740	-73.982580	2007.000000	2.000000	1.000000	0.220000	2.000000	1.000000	3.000000
50%	2.913660e+07	4.911774e+10	40.722290	-73.954440	2012.000000	3.000000	7.000000	0.740000	3.000000	1.000000	96.000000
75%	4.320120e+07	7.399650e+10	40.762760	-73.932350	2017.000000	5.000000	30.000000	2.000000	4.000000	2.000000	269.000000
max	5.736742e+07	9.876313e+10	40.916970	-73.705220	2022.000000	5645.000000	1024.000000	90.000000	5.000000	332.000000	3677.000000

Dall'analisi empirica si evince anche la presenza di valori mancanti o nulli che andranno gestiti.

Construction year	price	service fee	minimum nights	number of reviews	last review	reviews per month	review rate number	calculated host listings count	availability 365
2020.0	$966	$193	10.0	9.0	10/19/2021	0.21	4.0	6.0	286.0
2007.0	$142	$28	30.0	45.0	5/21/2022	0.38	4.0	2.0	228.0
2005.0	$620	$124	3.0	0.0	NaN	NaN	5.0	1.0	352.0
2005.0	$368	$74	30.0	270.0	7/5/2019	4.64	4.0	1.0	322.0
2009.0	$204	$41	10.0	9.0	11/19/2018	0.10	3.0	1.0	289.0

1.2 Pulizia e Formattazione dei dati

Sono stati individuati alcuni tipi di dato formattati in modo errato. Alcuni di loro necessitavano di essere formattati ad hoc, come per esempio la colonna 'price' e la colonna 'service fee' che erano immagazzinate come stringhe. Per un utilizzo corretto, sono stati trasformati in tipo numerico.

```
# Removing "$" symbol in 'price' column and converting values to numerical
self.data['price'] = self.data['price'].str.replace('$', '', regex=False)
self.data['price'] = self.data['price'].str.replace(',', '', regex=False)
self.data['price'] = self.data['price'].str.replace(' ', '', regex=False)
self.data['price'] = self.data['price'].astype('Int64')
# Same for column 'service fee'
self.data['service fee'] = self.data['service fee'].str.replace('$', '', regex=False)
self.data['service fee'] = self.data['service fee'].str.replace(',', '', regex=False)
self.data['service fee'] = self.data['service fee'].str.replace(' ', '', regex=False)
self.data['service fee'] = self.data['service fee'].astype(float)
```

Altri dati sono stati individuati come incongruenti con il proprio dominio e sono stati riportati a valori coerenti, come ad esempio i dati della colonna ‘availability 365’.

```
# Managing 'availability 365' column, changing all negative values to positive,
# then collapsing all values exceeding the maximum to 365
self.data['availability 365'] = npy.where(self.data['availability 365'] < 0,
                                          self.data['availability 365'] * -1,
                                          self.data['availability 365'])
self.data['availability 365'] = npy.where(self.data['availability 365'] > 365,
                                          365,
                                          self.data['availability 365'])
```

Sono stati individuati e corretti anche errori di tipo grammaticale ricorrenti nei dati di tipo stringa.

```
# Fixing specific typos in 'neighbourhood group' column
self.data['neighbourhood group'] = self.data['neighbourhood group'].replace('brookln', 'Brooklyn')
```

1.3 Riempimento dei valori nulli

Per la gestione dei valori nulli, si è condotta un’analisi specifica per ogni diversa colonna, considerato che ogni tipo di dato necessità di una strategia diversa per mantenerne l’integrità.

Alcuni dati mancanti sono stati riempiti con default values, per esempio

```
# Filling Na/Null values in 'host_identity_verified' column with the default value 'unconfirmed'
self.data['host_identity_verified'] = self.data['host_identity_verified'].fillna('unconfirmed')
```

Per altri è stato possibile calcolarli precisamente poiché dipendenti da altri dati

```
# Filling Na/Null values in 'calculated host listing count' column with the precise value
self.data['calculated host listings count'] = self.data.groupby('host id')[
    'calculated host listings count'].transform(lambda x: x.fillna(x.count()))
```

Essendo il dataset egualmente distribuito rispetto ai suoi dati, si è considerato lecito poter riempire dati cruciali con la media degli stessi, come ad esempio ‘price’ e ‘service fee’ , anche se questo metodo, seppur economico e semplice, può rivelarsi meno preciso di altre stime più complesse. In generale il metodo si è considerato efficace poiché la quantità di entries che ne erano affette era molto ridotta.

```
'''
   Considering the dataset is normally distributed, filling the holes with the mean is a good compromise
   of simplicity and cheapness, yet less precise and reliable than a more complex metric
'''
# Filling Na/NaN values in 'price' column with the mean value
self.data['price'] = self.data['price'].fillna(self.data['price'].mean().astype(int))
# Same for 'service fee' column
self.data['service fee'] = self.data['service fee'].fillna(self.data['service fee'].mean().astype(int))
```

Per altri valori si è stimato che la mediana fosse più adatta come valutazione

```
# Filling Na/Null values in the selected columns with the corresponding median or mean value,
# based on the consistency of the metric
self.data['number of reviews'] = self.data['number of reviews'].fillna(self.data['number of reviews'].median())
self.data['reviews per month'] = self.data['reviews per month'].fillna(self.data['reviews per month'].median())
self.data['availability 365'] = self.data['availability 365'].fillna(self.data['availability 365'].median())
self.data['review rate number'] = self.data['review rate number'].fillna(self.data['review rate number'].mean().astype(int))
```

Infine, per valori che non potevano essere stimati con precisione, tra i quali alcuni considerati estremamente importanti per mantenere l'integrità, si è deciso di eliminare direttamente le entries affette; ultimo passaggio è stato eliminare eventuali duplicati.

1.4 Analisi dei Dati

Fondamentale per l'obbiettivo del progetto è individuare la feature target su cui andare a calcolare previsioni. Nel caso specifico, la scelta è ricaduta sulla feature 'price', prezzo, in quanto elemento chiave delle valutazioni in questo campo.

Obbiettivo specifico è identificare quali fattori possano avere un impatto significativo sul prezzo di affitto, tra gli elementi presenti nel dataset. I macrogruppi di feature nel nostro caso comprendono, fra gli altri:

- Host-related attributes: come "host_identity_verified" e "host_response_rate".
- Property attributes: come "room_type", "accommodates", "bathrooms", "bedrooms".
- Location-related attributes: come "neighbourhood", "latitude", e "longitude".
- Review-related attributes: come "review_scores_rating" e "number_of_reviews".
- Booking attributes: come "availability_365", "minimum_nights", e "cancellation_policy".
- Country-related attributes: come "country" e "country_code"

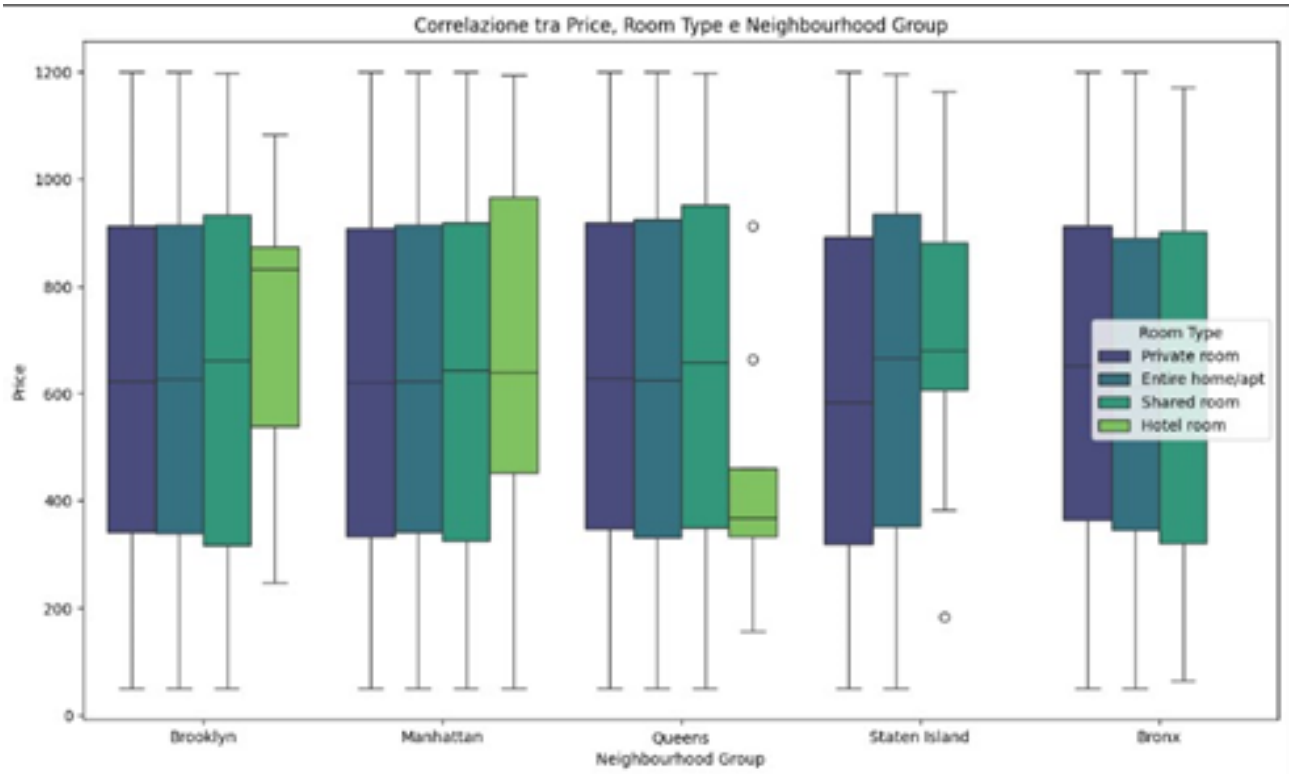
1.4.1 Ricerca di correlazioni con 'price'

In prima battuta, si è evidenziato che il dataset contiene solo riferimenti a proprietà nel territorio statunitense, in particolare nell'area urbana di New York. I dati presenti nelle colonne relative alla nazione possono essere quindi eliminati poiché non ci forniscono alcuna informazione.

Sono stati rilevati come poco significativi anche i dati relativi all'host della proprietà, individuati da valori come nome, id e numero di proprietà indirizzate sul sito.

Un'analisi della posizione delle proprietà rispetto ai valori di prezzo ha rivelato gli attributi di latitudine e longitudine come poco significativi in quanto le proprietà sono egualmente distribuite rispetto al prezzo, non evidenziando alcuna correlazione.

L'attributo 'neighbourhood group' può essere eliminato, poiché si tratta solamente di una generalizzazione rispetto all'attributo 'neighbourhood' ed è stata evidenziata una sostanziale uniformità dei prezzi tra i vari gruppi.



Diverso discorso invece per i quartieri di appartenenza, la cui analisi rispetto alla feature target ha rivelato correlazioni, anche in funzione del tipo di stanza, risultato che potevamo definire come atteso. Questo li qualifica positivamente per la feature selection.

Nonostante ci si aspetti che diverse ‘cancellation policy’ siano associate a variazioni di prezzo, l’analisi della media dei prezzi rispetti alle varie categorie ha evidenziato come sussista una naturale uniformità all’interno del dataset, candidando la colonna all’eliminazione.

Anche il dato del ‘availability 365’ si presenta come egualmente distribuito tra i dati, rispetto al prezzo. La sua eliminazione non influenzerà la predizione.

1.4.2 Feature Eliminate

In generale le feature eliminate sono le seguenti:

- License
- Country
- Country code
- Neighbourhood group
- Name
- Host name
- Host identity verified
- Host listing count
- House rules
- Cancellation policy
- Lat
- Long

Le feature rimanenti sono state scelte come significative per la predizione del ‘price’ delle proprietà: alcune di loro però sono in formato categorico e si è reso necessario un passaggio di encoding per poterne rappresentare in maniera numerica.

2. Apprendimento Supervisionato

L'apprendimento supervisionato è una tecnica di Machine Learning in cui un modello viene addestrato utilizzando un dataset etichettato, ovvero un insieme di dati in cui ogni esempio è associato a una risposta corretta. Durante l'addestramento, il modello apprende a mappare input specifici (input feature) alle corrispondenti etichette o output (target feature). L'obiettivo è che, una volta addestrato, il modello sia in grado di generalizzare e fare previsioni accurate su nuovi dati non visti.

Questo tipo di apprendimento è chiamato "supervisionato" poiché il processo è guidato dall'informazione fornita dalle etichette, che supervisionano e correggono le previsioni del modello. Gli algoritmi più comuni utilizzati in apprendimento supervisionato includono la Linear Regression, le Support Vector Machines (SVM), le Random Forests e le Artifical Neural Networks.

2.1 Train-Test split

L’insieme di input viene diviso in due parti: un training set e un test set.

Il Training Set è l’insieme di dati su cui il modello viene effettivamente addestrato. Il modello usa questi dati per imparare e adattarsi in modo da poter fare previsioni o classificazioni accurate.

Il Test Set è l’insieme di dati che viene utilizzato per valutare la precisione del modello. Non viene utilizzato durante la fase di addestramento, in modo da fornire al modello dati inediti su cui poter essere valutato.

Nel nostro caso, dopo dovute valutazioni, si è optato per una divisione 80-20 con l’80% degli esempi che comporrà il Training Set e il restante 20% comporrà il Test Set.

2.2 Scaling

Lo scaling nel machine learning è il processo di trasformazione delle feature di un dataset in modo che abbiano una scala comune. Questa operazione è cruciale per molti algoritmi poiché l'assenza di scaling può portare a prestazioni subottimali o addirittura a risultati errati.

In questo caso di studio si è optato per l'utilizzo del Min-Max Scaling

Min-Max Scaling:

- Trasforma i dati in un intervallo specifico, di solito $[0, 1]$.
- Formula: $X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$
- È particolarmente utile quando i dati devono essere mantenuti entro un intervallo specifico.

2.3 Scelta del Modello

Dato che la nostra feature target 'price' è rappresentata come valore numerico, il nostro obbiettivo è un task di regressione.

Per il caso di studio, sono stati testati 2 modelli di regressione lineare: il Random Forest e l'Extreme Gradient Boost.

- Random Forest è un algoritmo di machine learning basato su l'ensemble learning, che combina le previsioni di più alberi decisionali per migliorare la precisione e la robustezza del modello. Esso costruisce una "foresta" di alberi decisionali su sottoinsiemi casuali del dataset e utilizza la media dei risultati degli alberi per generare la previsione finale. Random Forest tende a non soffrire di overfitting poiché la combinazione di molti alberi aiuta a smussare le anomalie dei singoli modelli.
- Extreme Gradient Boosting (XGB) è un'implementazione ottimizzata del concetto di gradient boosting, ovvero combinare modelli deboli (solitamente alberi decisionali poco profondi) a cascata dove ogni nuovo modello cerca di correggere gli errori residui fatti dal modello precedente per formare un modello forte e accurato. Molto efficiente

e potente, è noto per essere estremamente veloce grazie all'uso di tecniche come la parallelizzazione, la riduzione della complessità computazionale e l'uso intelligente della memoria.

2.3.1 Validazione

Entrambi i modelli sono stati testati attraverso l'applicazione della K-Fold Cross Validation, una tecnica ampiamente utilizzata in machine learning per valutare la performance di un modello e prevenire l'overfitting. Consiste nel suddividere il dataset in k sottoinsiemi o folds di uguale dimensione. Il numero di fold k è scelto a seconda della dimensione del dataset e delle esigenze specifiche.

Con questo metodo, il modello viene addestrato k volte. In ogni iterazione, uno dei fold viene utilizzato come set di validazione, mentre gli altri k-1 fold vengono usati per addestrare il modello. Le prestazioni del modello vengono poi valutate per ciascuno dei k fold, e le metriche (ad esempio R-squared o il Mean Squared Error) vengono mediate per ottenere una stima complessiva della performance del modello.

In generale, la K-fold Cross Validation è una tecnica fondamentale per valutare in modo robusto le prestazioni di un modello, assicurando che esso sia in grado di generalizzare bene su dati nuovi e non visti.

Le metriche menzionate, R-squared e Mean Squared Error, sono state utilizzate per la nostra validazione.

L'R^2 (R-squared), noto anche come coefficiente di determinazione, è una metrica di valutazione comunemente utilizzata per misurare la bontà dell'adattamento del modello ai dati osservati; rappresenta la proporzione della varianza nella target feature che è spiegata dal modello rispetto alla features in input.

Nell'interpretazione di R^2:

1.0: Indica un modello perfetto, dove il 100% della varianza dei dati è spiegata dal modello.

0.0: Indica che il modello non spiega nessuna della varianza nei dati rispetto alla media.

Valori Negativi: Un R^2 negativo può verificarsi quando il modello è peggiore nel predire rispetto alla media, ad esempio, se non include una costante o è mal progettato.

La formula per calcolare R^2 è la seguente:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Dove:

- y_i sono i valori osservati.
- $\hat{y}_i$ sono i valori predetti dal modello.
- $\bar{y}$ è la media dei valori osservati.

Mentre, Il Mean Squared Error (MSE) è un'altra metrica di valutazione comunemente utilizzata. È calcolata come la media dei quadrati degli errori, dove l'errore è la differenza tra il valore osservato (reale) e il valore predetto dal modello. Il MSE penalizza gli errori grandi più degli errori piccoli, per via dell'utilizzo del quadrato, e ciò che lo rende particolarmente sensibile agli outlier, che possono influenzare negativamente il punteggio.

Nell'interpretazione del MSE, un valore più basso rispetto a quello individuato su un altro modello indica una precisione più alta e migliori performance.

La formula del MSE è la seguente:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Dove:

- n è il numero totale delle osservazioni.
- y_i rappresenta i valori osservati.
- $\hat{y}_i$ sono i valori predetti dal modello.

2.3.2 Risultati della Validazione

Tra i due modelli analizzati, il Random Forest ha ottenuto i risultati migliori, in generale molto promettenti per il nostro task.

Gli iperparametri che hanno ottenuto i risultati migliori sono:

***n_estimators:* 100** Per controllare il numero di alberi predittori nella foresta

random_state: 42 Per stabilire la randomicità nel bootstrap sampling e nel best split sampling

Di seguito vengono mostrati i risultati di una 10-fold Cross Validation effettuata con il Random Forest impostato con i suddetti iperparamentri.

```
K-Fold Cross-Validation Results (k=10):
Mean Squared Error on average: 524.6866 (+/- 245.1136)
R-squared Score on average:    0.9952   (+/- 0.0022)

Individual Fold Scores:
Fold 1:  MSE = 746.2955, R2 = 0.9931
Fold 2:  MSE = 366.7123, R2 = 0.9967
Fold 3:  MSE = 464.6054, R2 = 0.9957
Fold 4:  MSE = 500.4652, R2 = 0.9955
Fold 5:  MSE = 514.4315, R2 = 0.9953
Fold 6:  MSE = 402.3072, R2 = 0.9963
Fold 7:  MSE = 571.9092, R2 = 0.9948
Fold 8:  MSE = 447.8126, R2 = 0.9960
Fold 9:  MSE = 489.7091, R2 = 0.9955
Fold 10: MSE = 742.6177, R2 = 0.9933
```

Notiamo subito che R^2 mantiene valori estremamente ottimali su tutti i fold, mentre MSE ha più varianza ma accettabile considerando la sensibilità della metrica agli outlier.

2.3.3 Risultati della Predizione

Il modello è stato quindi addestrato su tutto il dataset con gli iperparametri individuati e successivamente testato con il Test Set ricavato in precedenza. È stato creato un grafico di dispersione per visualizzare la relazione tra i prezzi reali nel Test Set e quelli calcolati dal predittore.

Considerato che il predittore ha stimato circa 20000 risultati, le performance sembrano eccellenti, con solo una manciata di valori che sono stati erroneamente predetti erroneamente, in particolare sulla direzione della media. Se il modello fosse affetto da Overfitting, quest'ultimi valori rappresenterebbero la maggioranza.

Le metriche dei risultati ottenuti sono

```
Final Model Performance (trained on entire dataset):

Mean Squared Error: 590.0057
R-squared Score:      0.9946

Standard Deviation of Predicted Values: 330.8947
Standard Deviation of Actual Values:    331.6778
```

Con un piccolo peggioramento rispetto a quelle ottenute sul Training Set con 10-fold.

3. Conclusioni

Il modello di RandomForestRegressor ha fornito buone prestazioni con un MSE ridotto e un valore di R^2 estremamente vicino alla perfezione, dimostrando la sua capacità di prevedere accuratamente i prezzi degli annunci Airbnb sulla base delle caratteristiche fornite.

3.1 Possibili Miglioramenti

- **Feature Engineering**: Potrebbero essere create nuove caratteristiche, come la stagionalità, per migliorare ulteriormente le prestazioni del modello.
- **Hyperparameter Tuning**: Un tuning più approfondito degli iperparametri del modello potrebbe migliorare ulteriormente le prestazioni.
- **Modelli Alternativi**: Altri modelli come le Reti Neurali potrebbero essere sperimentati per verificare se forniscono migliori risultati, ed utilizzate i risultati per task predittive più avanzate. Oppure si potrebbe provare a predire la competitività sul mercato di una proprietà grazie alla tecnica di apprendimento non supervisionato del Clustering.