

Relazione Big Data

Progetto n. 9: Hotel Reviews

Giuseppe Pasquale Caligiure - Mat. 280867

2026

Indice

1	Presentazione del Progetto	2
1.1	Contesto di lavoro e obiettivi Realizzati	2
1.2	Architettura Frontend/Backend	2
1.3	Tecnologie Utilizzate e Requisiti	2
1.4	Logica di Funzionamento	3
2	Descrizione del Dataset Hotel_Reviews	3
2.1	Campi del Dataset	3
3	Descrizione delle Query Implementate	4
3.1	Trend Recensioni (Time Series)	4
3.2	Analisi Influenza Tag	6
3.3	Analisi Bias Nazionalità	8
3.4	Analisi Competitività Locale	10
3.5	Segmentazione Hotel (K-Means Clustering)	12
3.6	Migliori Hotel per Nazione	15
3.7	7. Locali vs Turisti	17
3.8	8. Preferenze Stagionali	17
3.9	9. Analisi Durata Soggiorno	17
3.10	10. Esperienza del Recensore	17
4	Conclusioni Finali	17

1 Presentazione del Progetto

1.1 Contesto di lavoro e obiettivi Realizzati

Il presente progetto è stato realizzato lavorando sul dataset “Hotel Reviews”, contenente oltre 515.000 recensioni di alberghi di lusso europei. L’applicativo realizzato consente di effettuare interrogazioni aggregate sul dataset, con l’obiettivo di estrarre insight significativi dai dati. Sfruttando le potenzialità di elaborazione distribuita offerte dal framework Spark, sono stati realizzati diversi moduli di analisi che consentono di rilevare aspetti temporali, testuali, geospaziali e comportamentali nelle recensioni del dataset. Gli obiettivi principali raggiunti includono:

- Identificazione dei trend di gradimento degli hotel nel tempo.
- Analisi dell’influenza di specifici tag (caratteristiche del soggiorno) sul punteggio finale.
- Segmentazione geografica e analisi della competitività locale.
- Profilazione degli hotel tramite algoritmi di Machine Learning (Clustering).
- Studio delle preferenze in base alla nazionalità e tipologia di viaggiatore.

1.2 Architettura Frontend/Backend

L’applicazione è stata realizzata seguendo una logica Frontend/Backend:

- **Backend (Spark):** Il file `queries.py` contiene la logica di lavoro. Ogni funzione implementa una diversa analisi dei dati, ma tutte rispettano il seguente schema: accetta un DataFrame Spark in input e restituisce un DataFrame Spark trasformato con i risultati.
- **Frontend (Streamlit):** Il file `app.py` gestisce l’interfaccia utente. All’avvio inizializza una `SparkSession` (cachata per efficienza) e carica il dataset. Quando l’utente seleziona un’analisi, il frontend invoca la funzione corrispondente dal backend, converte i risultati aggregati (di dimensioni ridotte) in Pandas DataFrame e li visualizza tramite grafici e tabelle.

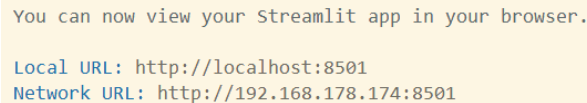
1.3 Tecnologie Utilizzate e Requisiti

Il progetto è stato sviluppato in Python, utilizzando il framework di Spark per eseguire interrogazioni in maniera scalabile e distribuita.

- **Linguaggio:** Python 3.11. (il progetto è stato testato con Python 3.11.9)
- **Backend: Apache Spark** (PySpark) per l’elaborazione parallela e distribuita dei dati. In particolare sono stati utilizzati DataFrame Spark, Spark SQL, Window Functions e User Defined Functions (UDF).
- **Machine Learning: Spark MLlib** per operazioni di clustering (K-Means) e **Scikit-learn** per regressioni lineari all’interno di UDF pandas.
- **Frontend: Streamlit** per la creazione di una web-app interattiva che permette all’utente di eseguire query e filtrare risultati.
- **Visualizzazione: Altair, PyDeck e Pandas** per la creazione di grafici interattivi e mappe geospaziali.
- **Gestione Dipendenze Windows:** Winutils e Hadoop per l’esecuzione locale su ambiente Windows.

1.4 Logica di Funzionamento

1. L'utente avvia l'applicazione tramite script batch (`RUN_APP_Hotel_Reviews.bat`) o comando Streamlit (`python -m streamlit run app.py`).
2. L'app si avvia automaticamente in una finestra del browser predefinito del dispositivo in uso, ma è anche raggiungibile da altri dispositivi (pc, tablet, smartphone) collegati sulla stessa rete locale, tramite gli indirizzi specificati nel terminale (Figura 1).



```
You can now view your Streamlit app in your browser.  
  
Local URL: http://localhost:8501  
Network URL: http://192.168.178.174:8501
```

Figura 1: URL della web-app

3. Dopo il caricamento iniziale del dataset in memoria (DataFrame Spark), tramite una sidebar laterale è possibile selezionare una delle query disponibili.
4. Ogni query espone parametri specifici (es. numero minimo di recensioni, raggio in km) modificabili tramite slider o input box. L'esecuzione della query avviene on-demand sfruttando il motore Spark. I risultati vengono visualizzati in-app tramite grafici, tabelle e mappe interattive.

2 Descrizione del Dataset Hotel_Reviews

Il dataset utilizzato è `Hotel_Reviews.csv` (reperibile su: <https://www.kaggle.com/datasets/jiashenliu/515k-hotel-reviews-data-in-europe>). Questo archivio contiene oltre 515.000 recensioni di hotel di lusso in Europa, raccolte dal sito Booking.com, dove sono pubblicamente accessibili. Ogni riga del dataset corrisponde ad una recensione e presenta 17 campi che descrivono sia le caratteristiche dell'hotel, sia l'esperienza del cliente.

- **Dimensione File:** Circa 238 MB
- **Numero di Righe:** 515.738

2.1 Campi del Dataset

- **Hotel_Address:** Indirizzo dell'hotel.
- **Additional_Number_of_Scoring:** Numero di valutazioni aggiuntive (clienti che hanno lasciato solo una valutazione numerica dell'hotel, senza recensione).
- **Review_Date:** Data in cui è stata rilasciata la recensione.
- **Average_Score:** Punteggio medio storico dell'hotel (calcolato su tutte le recensioni ricevute dall'hotel nell'ultimo anno).
- **Hotel_Name:** Nome della struttura.
- **Reviewer_Nationality:** Nazionalità dell'utente che ha lasciato la recensione.
- **Negative_Review:** Testo del commento negativo ("No Negative" se assente).
- **Review_Total_Negative_Word_Counts:** Conteggio parole commento negativo.
- **Total_Number_of_Reviews:** Totale recensioni ricevute dall'hotel.

- **Positive_Review**: Testo del commento positivo ("No Positive" se assente).
- **Review_Total_Positive_Word_Counts**: Conteggio parole commento positivo.
- **Total_Number_of_Reviews_Reviewer_Has_Given**: Numero di recensioni rilasciate dell'utente in passato.
- **Reviewer_Score**: Voto assegnato dal recensore all'hotel.
- **Tags**: Lista di stringhe che descrivono il soggiorno (es. "Leisure trip", "Couple", "Stayed 2 nights").
- **days_since_review**: Giorni trascorsi fra la pubblicazione e lo scraping della recensione.
- **lat**: Latitudine dell'hotel.
- **lng**: Longitudine dell'hotel.

3 Descrizione delle Query Implementate

3.1 Trend Recensioni (Time Series)

Obiettivo: Analizzare il **trend temporale** dei punteggi degli hotel (utilizzando la Regressione Lineare) per identificare quali alberghi stanno migliorando o peggiorando nel tempo.

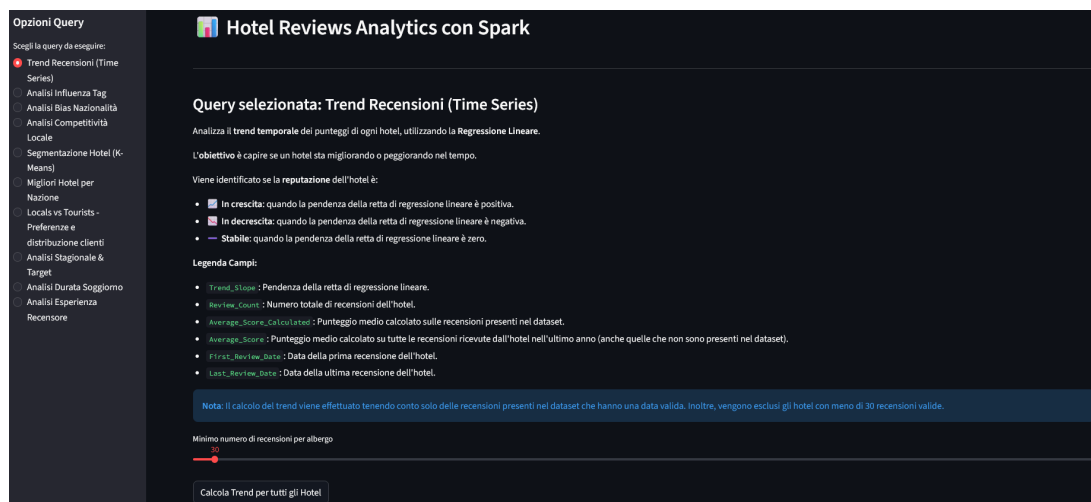


Figura 2: Query Trend Recensioni (Time Series)

Logica Backend:

- Le recensioni vengono **raggruppate per hotel** e ordinate cronologicamente.
- Per ogni gruppo (stesso hotel): da ogni recensione si estraggono i valori di **Reviewer_Score** e **Review_Date** (convertita in ordinale), quindi viene applicata una **regressione lineare (score vs tempo)** per calcolare la pendenza (**slope**) del **trend**. Inoltre, si effettua il calcolo di **altri valori aggregati**: punteggio medio, numero di recensioni, data della prima recensione, data dell'ultima recensione.
- Viene restituito un dataframe contenente i risultati ottenuti per ogni hotel.

Tecnologie: Viene definita una Pandas UDF (User Defined Function) per l'esecuzione della regressione lineare con `scikit-learn`. La UDF viene eseguita con la funzione `applyInPandas` affinché questa operazione sia parallelizzabile su ogni gruppo di hotel distribuito nei nodi Spark.

Risultati: Viene visualizzata la lista dei **Top 10 Hotel in Crescita** e dei **Top 10 Hotel in Calo**, ordinati per trend slope (Figura 3), dove trend positivo indica miglioramento, mentre trend negativo indica peggioramento. Inoltre, viene visualizzato il grafico **Distribuzione Trend vs Punteggio Medio** (Figura 4) che consente di individuare visivamente gli alberghi migliori o peggiori e leggerne le caratteristiche.

Casi d'uso: Identificare "stelle nascenti" o hotel decadenti nonostante un alto punteggio medio storico.

Hotel_Name	Trend_Slope	Review_Count	Average_Score_Calculated	Average_Score	First_Review_Date	Last_Review_Date
1076 The Curtain	0.0064	30	8.8833	9.1	2017-05-25	2017-08-03
931 Chasse Hotel	0.0059	139	9.0144	8.9	2017-03-27	2017-08-02
125 Hotel Park Lane Paris	0.0041	154	8.6338	8.7	2016-05-23	2017-08-03
1096 Villa Lut ce Port Royal	0.004	47	6.3851	7	2015-08-15	2017-05-02
853 NIX Milan	0.0035	180	8.5956	8.8	2017-02-25	2017-08-03
552 Hotel Capitol Milano	0.0031	66	8.1242	8.3	2015-09-22	2017-07-05
1298 Lansbury Heritage Hotel	0.0031	40	9.5175	9.4	2017-04-25	2017-08-02
1136 Best Western Le 18 Paris	0.0031	86	7.4291	7.6	2016-03-22	2017-07-20
265 Bob Hotel by Elegancia	0.003	49	8.9388	9	2017-03-12	2017-08-02
90 Hilton London Euston	0.0025	470	7.1209	7.4	2015-08-04	2017-08-02

Hotel_Name	Trend_Slope	Review_Count	Average_Score_Calculated	Average_Score	First_Review_Date	Last_Review_Date
632 Okko Hotels Paris Porte De Versailles	-0.0082	43	9.0837	9.2	2017-06-09	2017-07-31
1019 Le Tsuba Hotel	-0.0067	57	9.0807	9.3	2017-03-25	2017-08-02
380 Marlin Waterloo	-0.0062	84	8.3548	8.6	2017-05-29	2017-08-03
917 Arthotel ANA Westbahn	-0.0049	67	8.4418	8.3	2017-03-02	2017-08-03
1025 Maison Albar Hotel Paris C line	-0.0047	62	9.0065	8.9	2016-12-25	2017-08-03
1297 La Villa Haussmann	-0.0046	41	9.1561	9.2	2017-01-10	2017-08-02
1314 Majestic Hotel Spa	-0.0045	38	8.4842	8.7	2015-08-18	2017-07-29
904 Villa Eugenie	-0.0044	62	5.8645	6.8	2015-08-05	2017-08-02
830 London Suites	-0.0044	114	7.3211	7.4	2016-07-03	2017-08-02
183 Park Plaza London Park Royal	-0.004	197	8.901	8.9	2017-03-10	2017-08-03

Figura 3: Top 10 Hotel in Crescita/Calo

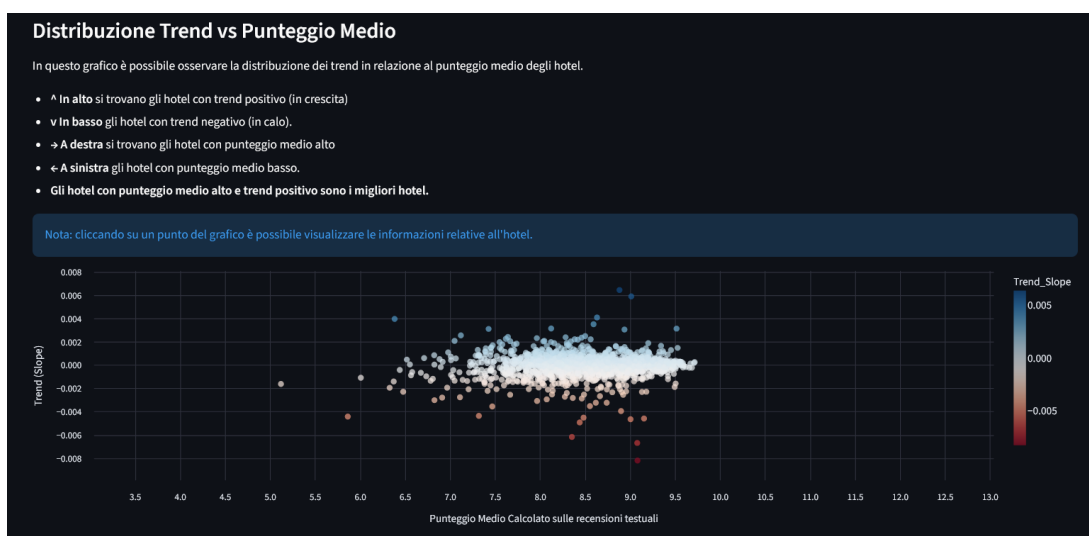


Figura 4: Distribuzione Trend vs Punteggio Medio

3.2 Analisi Influenza Tag

Obiettivo: Determinare quali fattori (es. "Single Room", "No Window") impattano positivamente o negativamente sul punteggio che i recensori assegnano agli hotel.

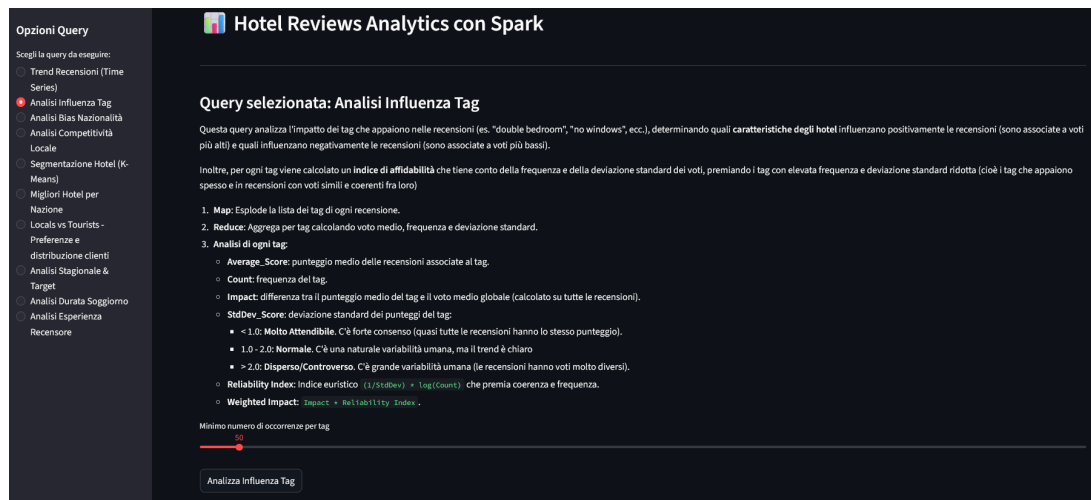


Figura 5: Query Analisi Influenza Tag

Logica Backend:

- Le recensioni vengono "esplose" seguendo la logica di una **FlatMap**: si prende in input la stringa dei tag di ogni recensione, la stringa viene suddivisa in singoli tag, ciascuno dei quali viene poi "ripulito" da eventuali spazi vuoti o lettere maiuscole, e infine si produce in output una riga per ogni singolo tag letto in input (gli altri campi della riga vengono duplicati dalla recensione originale da cui è stato estratto il tag).
- Le righe risultanti vengono raggruppate per tag, eseguendo delle operazioni di **aggregazione** sugli hotel associati ad ogni tag: conteggio del numero di hotel, calcolo del punteggio medio degli hotel e calcolo della **deviazione standard** (ad esempio, se la deviazione standard è alta, significa che i voti sono molto dispersi, quindi la valutazione del peso di quel tag sarà meno affidabile).
- Viene calcolata la **media globale** sui punteggi medi di tutti i tag, per poter avere un indice di confronto sulla base del quale valutare se un tag ha un **impatto** positivo o negativo rispetto agli altri (ad esempio, se un tag ha una media di 9.0 e la media globale è 8.5, allora quel tag ha un "impatto positivo" (+0.5)).
- Per ogni tag, vengono calcolati:
$$\text{Impact} = \text{Average_Score} - \text{Global_Average}$$
$$\text{Reliability_Index} = (1 / (\text{StdDev} + 0.1)) * \log(\text{Count})$$
$$\text{Weighted_Impact} = \text{Impact} * \text{Reliability_Index}$$

Il **Reliability_Index** è un indice euristico che valuta quanto è attendibile l'impatto di un tag, premiando i tag con una maggiore stabilità dei voti (deviazione standard bassa) e con un'alta frequenza (il logaritmo serve per mitigare l'impatto dei tag estremamente frequenti).
- Viene restituito un dataframe contenente i risultati ottenuti per ogni tag.

Risultati: Vengono visualizzate le classifiche dei **tag con il maggior impatto positivo e negativo** (Figura 6), ordinati per **Weighted_Impact** decrescente/crescente. Inoltre, viene visualizzato il grafico **Affidabilità vs Impatto** (Figura 7), che consente di individuare i fattori che influenzano maggiormente (in positivo o in negativo) la valutazione degli hotel e valutare a colpo d'occhio il grado di attendibilità di ciascuno di essi.

Casi d'uso: Identificare le **caratteristiche più apprezzate o criticate** dai clienti degli hotel.



Figura 6: Top 10 Tag Positivi/Negativi



Figura 7: Grafico Affidabilità vs Impatto

3.3 Analisi Bias Nazionalità

Obiettivo: Individuare, se esistono, le nazionalità che tendono a dare voti più alti o bassi rispetto alla media.



Figura 8: Query Analisi Bias Nazionalità

Logica Backend:

- Raggruppamento delle recensioni per **Reviewer_Nationality**.
- Calcolo di valori aggregati per ogni nazionalità: numero di recensioni, media dei punteggi assegnati nelle recensioni, media del numero di parole positive scritte nelle recensioni, media del numero di parole negative scritte nelle recensioni.
- Calcolo della **media globale** di tutti i punteggi assegnati in tutte le recensioni (da utilizzare per confronto).
- Calcolo dello **Score_Deviation** di ogni nazionalità, cioè la differenza fra la media dei punteggi assegnati da recensori di quella nazionalità e la media globale. Questa metrica indica se mediamente i recensori di una certa nazionalità sono più **critici (deviazione negativa)** o **generosi (deviazione positiva)** nell'assegnare le valutazioni degli hotel.
- Calcolo del **Sentiment_Ratio** di ogni nazionalità, cioè il rapporto fra numero di parole positive e negative utilizzate nelle recensioni. Questa metrica, quando considerata in relazione con il punteggio medio assegnato dai recensori di una certa nazionalità, indica la **coerenza** di questi recensori: chi dà più spesso valutazioni numeriche positive dovrebbe avere un **Sentiment_Ratio** positivo (cioè dovrebbe utilizzare mediamente più parole positive che negative), mentre chi è più critico e dà più spesso valutazioni numeriche negative dovrebbe avere un **Sentiment_Ratio** negativo.
- Restituzione del dataset aggregato per **Reviewer_Nationality**, con le metriche calcolate.

Risultati: Vengono visualizzate le classifiche delle **nazionalità più critiche e più generose** (Figura 9), ordinate per **Score_Deviation** crescente/decescente. Inoltre, viene visualizzato il grafico **Correlazione Voto vs Positività Testo** (Figura 10), che consente di valutare visivamente la coerenza dei recensori per ogni nazionalità.

Casi d'uso: Profilazione per nazionalità e identificazione di bias culturali che peggiorano/migliorano l'esperienza dei clienti.

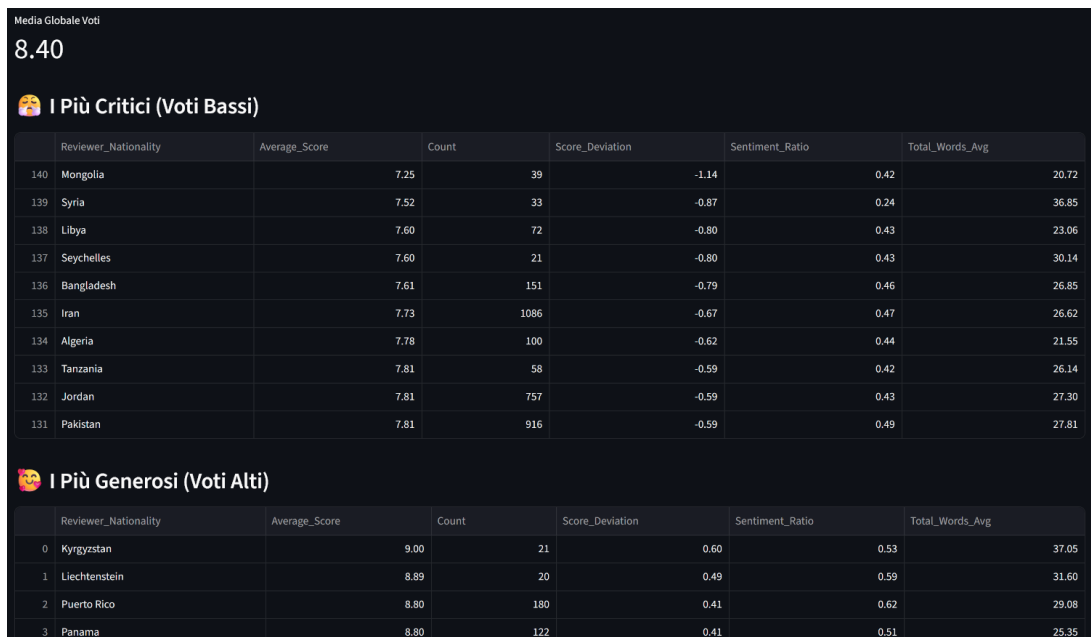


Figura 9: Classifica nazionalità più critiche e più generose

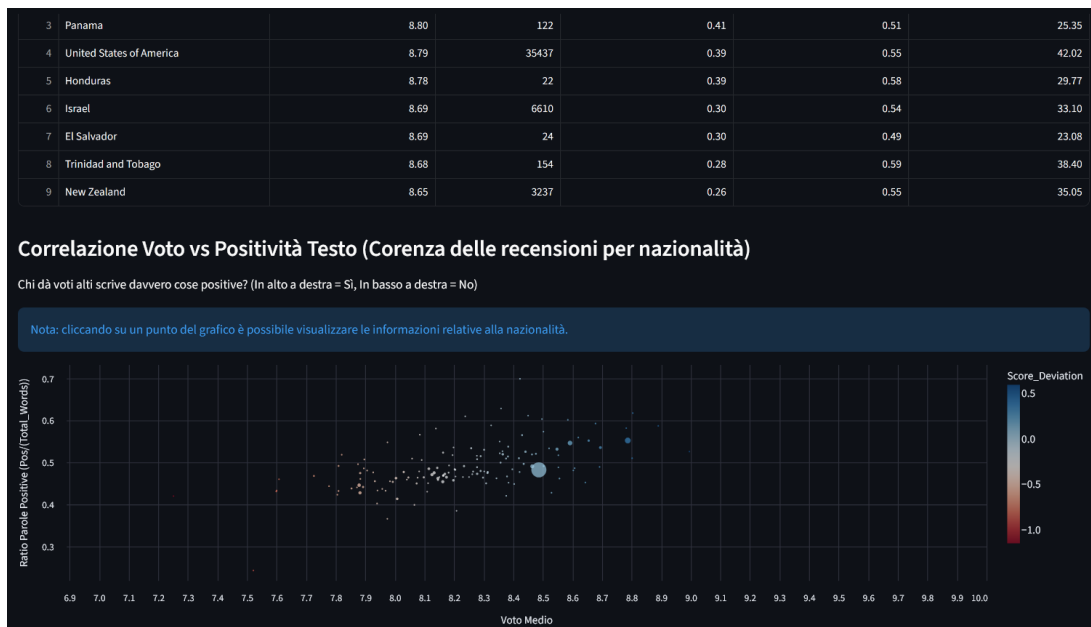


Figura 10: Grafico Correlazione Voto vs Positività Testo

3.4 Analisi Competitività Locale

Obiettivo: Confrontare le performance di un hotel rispetto ai suoi concorrenti geograficamente più vicini.

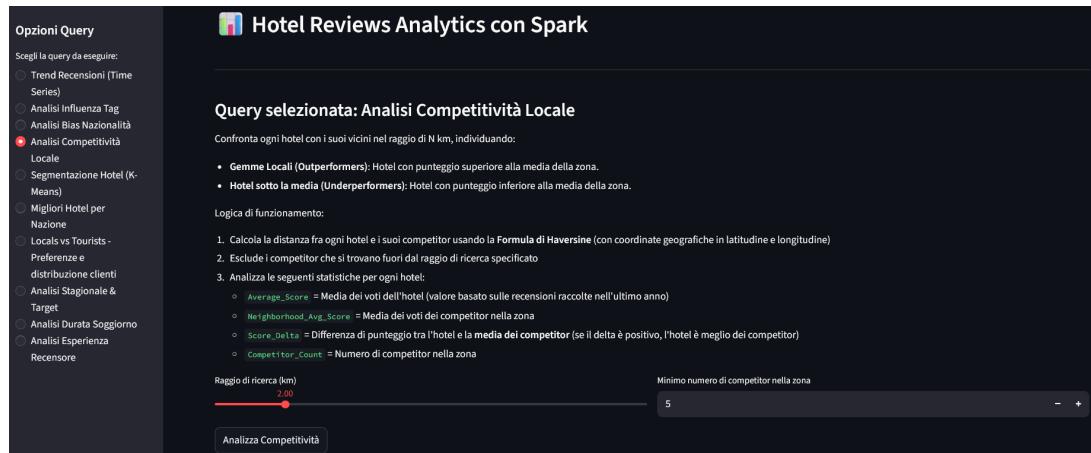


Figura 11: Query Analisi Competitività Locale

Logica Backend:

- Esegue un **self-join** del dataset per calcolare tutte le possibili coppie di hotel distinti: `Hotel_A` e `Hotel_B`.
- Calcola la **distanza geografica fra ogni coppia di hotel** (utilizzando la Formula di Haversine).
- Filtra le coppie di hotel, mantenendo solo quelle che hanno una distanza inferiore a `N` chilometri (`N` è un parametro scelto dall'utente).
- Raggruppa per `Hotel_A`, ottenendo per ogni hotel il suo vicinato. Poi aggrega i campi degli hotel appartenenti al vicinato, calcolando **punteggio medio del vicinato** e **numero di concorrenti**. Infine, calcola `Score_Delta` = differenza fra il punteggio di `Hotel_A` e la media del vicinato.
- Classifica come **Outperformer** gli hotel con `Score_Delta > 0` e come **Underperformer** quelli con `Score_Delta < 0`. Infine, restituisce il dataset con i valori aggregati appena calcolati.

Risultati: Vengono visualizzate le classifiche delle **Gemme Locali** (i migliori **Outperformer**) e dei **peggiori Underperformer** (Figura 12), ordinate per `Score_Delta` decrescente/crescente. Inoltre, viene visualizzato il grafico **Performance Relativa** (Figura 13), che consente di individuare visivamente gli hotel che appartengono all'una o all'altra categoria, confrontando il voto dell'hotel con il voto medio del suo vicinato.

Casi d'uso: Analisi della competitività fra attività geograficamente vicine.

💎 Top 10 Gemme Locali (Meglio dei competitor vicini)					
	Hotel_Name	Average_Score	Neighborhood_Avg_Score	Score_Delta	Competitor_Count
0	Milestone Hotel Kensington	9.50	8.23	1.27	100
1	Ritz Paris	9.80	8.59	1.21	230
2	Nu Hotel	8.90	7.82	1.08	5
3	Hotel The Peninsula Paris	9.50	8.44	1.06	129
4	Waldorf Astoria Amsterdam	9.50	8.48	1.02	65
5	Haymarket Hotel	9.60	8.59	1.01	134
6	Pillows Anna van den Vondel Amsterdam	9.40	8.39	1.01	47
7	Hotel Eiffel Blomet	9.40	8.39	1.01	46
8	Charlotte Street Hotel	9.50	8.49	1.01	136
9	H tel de La Tamise Esprit de France	9.60	8.61	0.99	228

⚠️ Top 10 Sotto la Media (Peggio dei competitor vicini)					
	Hotel_Name	Average_Score	Neighborhood_Avg_Score	Score_Delta	Competitor_Count
1376	Hotel Liberty	5.20	8.42	-3.22	41
1375	Hotel Cavendish	6.40	8.53	-2.13	120
1374	The Tophams Hotel	6.60	8.71	-2.11	88
1373	Savoy Hotel Amsterdam	6.40	8.35	-1.95	41
1372	Britannia International Hotel Canary Wharf	7.10	8.96	-1.86	8
1371	Bloomsbury Palace Hotel	6.80	8.53	-1.73	119

Figura 12: Classifica migliori Outperformer e peggiori Underperformer

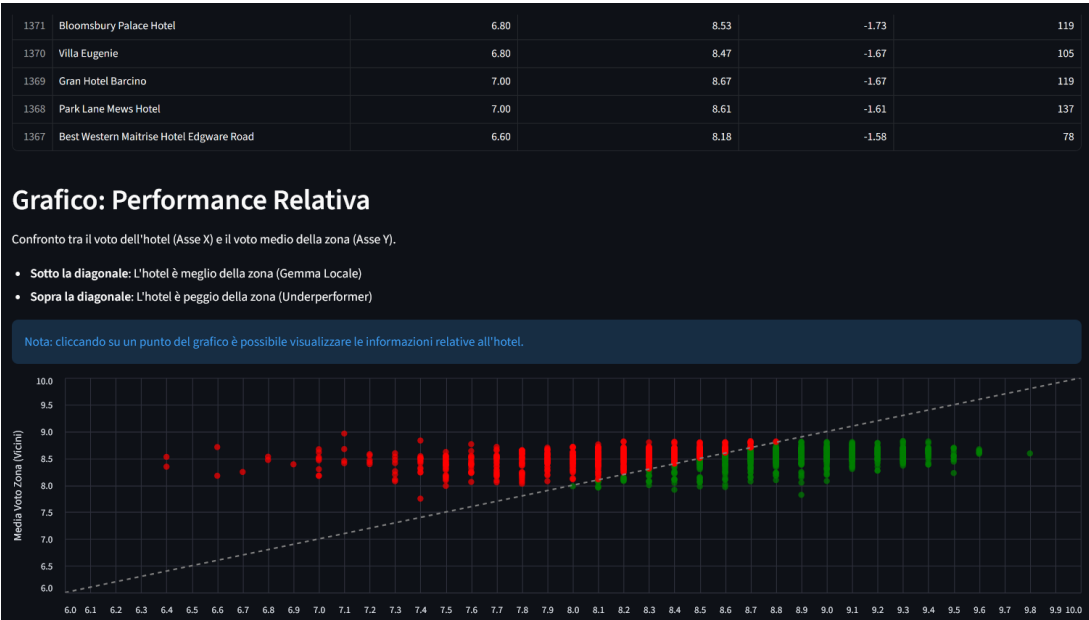


Figura 13: Grafico Performance Relativa

3.5 Segmentazione Hotel (K-Means Clustering)

Obiettivo: Raggruppare gli hotel in cluster omogenei (utilizzando K-Means Clustering), sulla base delle seguenti feature: Punteggio Medio, Popolarità (numero recensioni), Verbosità delle recensioni, Posizione geografica, Bias di nazionalità. L'utente seleziona fra queste le caratteristiche sulla base delle quali saranno raggruppati gli hotel. Inoltre, può scegliere il numero di gruppi in cui effettuare la suddivisione.



Figura 14: Query Segmentazione Hotel (K-Means Clustering)

Logica Backend:

- Raggruppamento per hotel e aggregazione per il calcolo delle feature.
- Creazione di vettore delle feature con VectorAssembler e normalizzazione con StandardScaler.
- Clustering degli hotel con algoritmo K-Means.

Tecnologie: Da Spark MLlib: VectorAssembler, StandardScaler, KMeans, Pipeline).

Risultati: Assegnazione di ogni hotel a un cluster (es. "Hotel Popolari di Lusso", "Hotel Economici di Nicchia").

- Analisi delle caratteristiche di ogni cluster (Figura 15).
- Mappa interattiva della distribuzione geografica dei cluster (Figura 16).
- Grafico interattivo per l'esplorazione delle performance dei cluster, con possibilità di confrontare: Punteggio medio, Numero di recensioni, Numero medio di parole positive nelle recensioni, Numero medio di parole negative nelle recensioni (Figura 17).
- Grafico della distribuzione delle nazionalità dei cluster (Figura 18).

Casi d'uso: Segmentazione di marketing, raccomandazioni di hotel simili.

Analisi dei Gruppi Identificati

	Cluster Name	Count	Avg_Score	Total_Reviews	Avg_Pos_Words	Avg_Neg_Words
0	Cluster 0	207	8.50	1425	19.5	18.3
1	Cluster 1	844	8.50	800	18.8	17.2
2	Cluster 2	297	8.43	1418	18.4	17.5
3	Cluster 3	127	8.28	4124	17.2	19.6

Figura 15: Analisi delle caratteristiche di ogni cluster

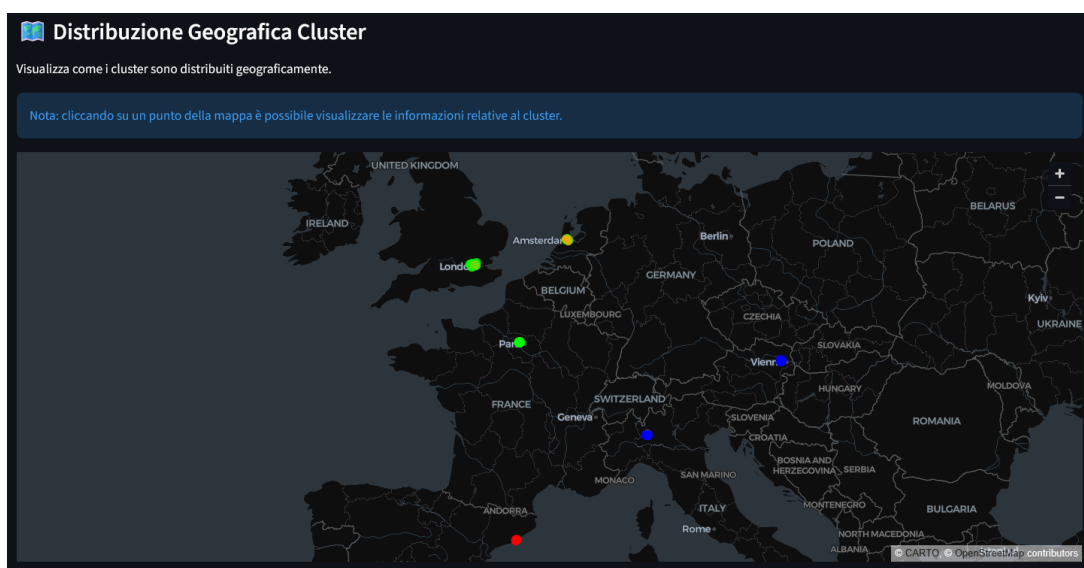


Figura 16: Mappa interattiva della distribuzione geografica dei cluster

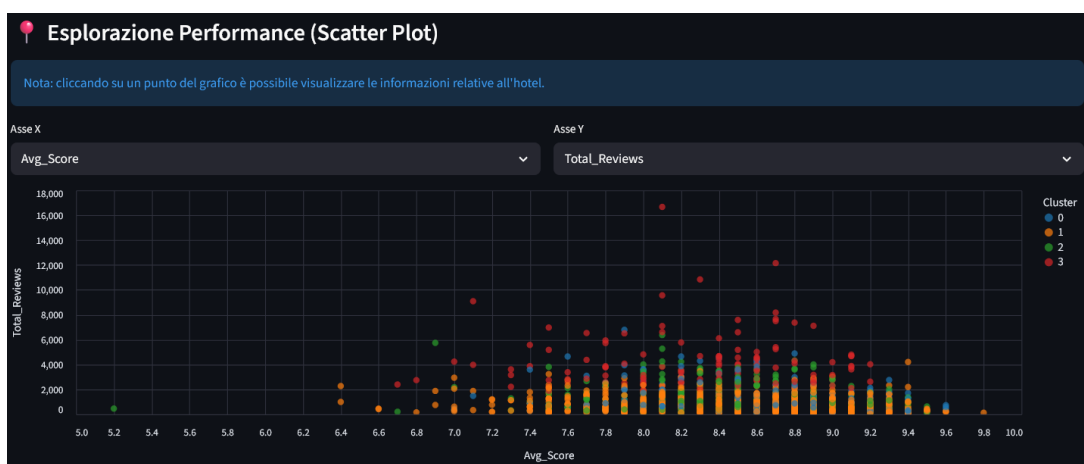


Figura 17: Grafico interattivo per l'esplorazione delle performance dei cluster

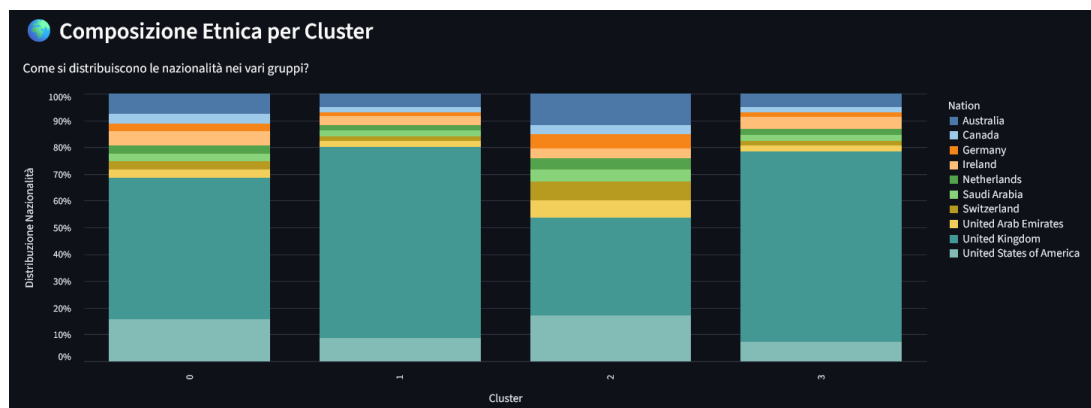


Figura 18: Grafico della distribuzione delle nazionalità dei cluster

3.6 Migliori Hotel per Nazione

Obiettivo: Identificare le eccellenze alberghiere suddivise per paese.



Figura 19: Query Migliori Hotel per Nazione

Logica Backend:

- Deduplicazione degli hotel nelle recensioni.
- Estrazione della nazione dall'indirizzo dell'hotel.
- Raggruppamento delle strutture per nazione e ordinamento in base al punteggio medio e al numero di recensioni (in caso di parità).

Risultati: Viene visualizzata la lista dei top N hotel per ogni nazione presente nel dataset (le nazioni sono presentate in ordine alfabetico) e la distribuzione dei punteggi dei migliori hotel in assoluto (Figura 20). Inoltre, viene visualizzata la mappa geografica che mostra dove si trovano gli hotel (Figura 21).

Casi d'uso: Utenti che cercano i migliori hotel in una specifica destinazione turistica.

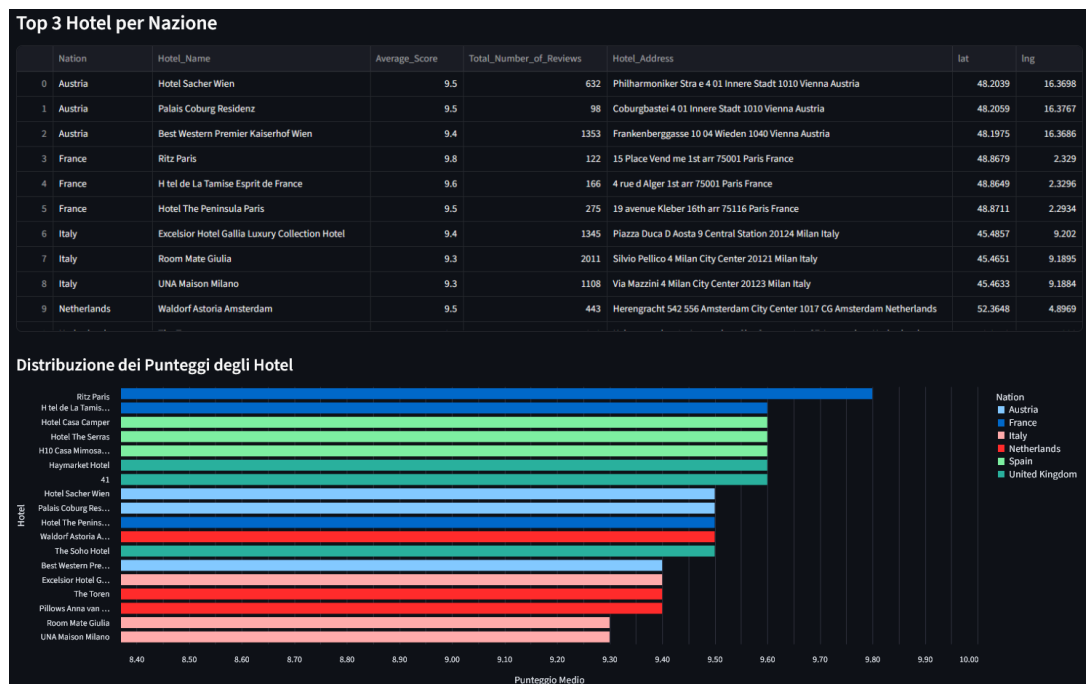


Figura 20: Top N Hotel per Nazione e Distribuzione dei Punteggi degli Hotel

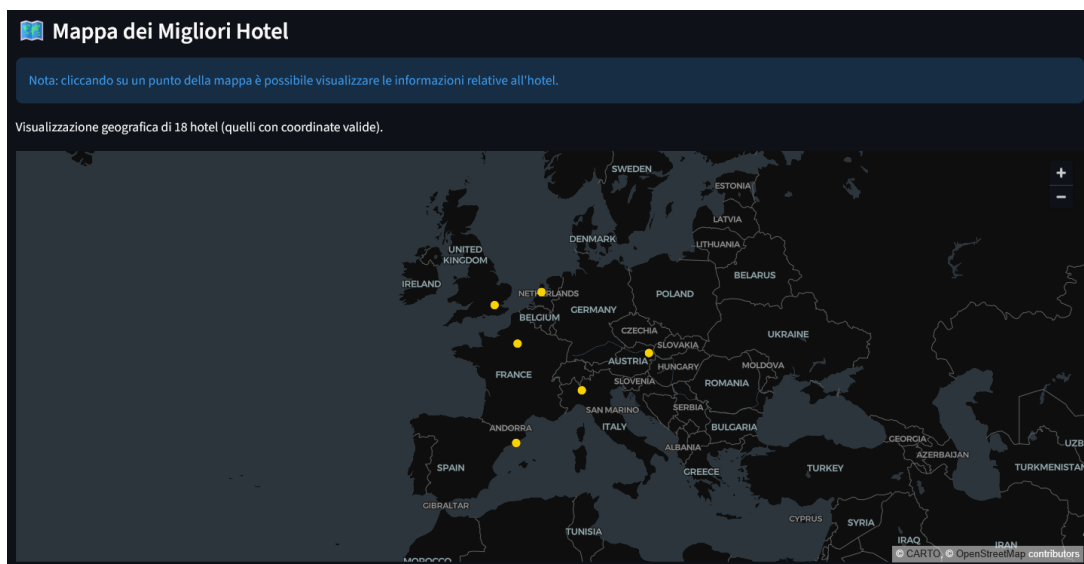


Figura 21: Mappa dei Migliori Hotel

3.7 7. Locali vs Turisti

Obiettivo: Analizzare la differenza di percezione tra chi visita il proprio paese (Local) e chi viene dall'estero (Tourist).

Logica: Confronto tra la nazione dell'hotel e la nazione del recensore. Calcolo separato delle medie per i due gruppi.

Tecnologie: Conditional Aggregation.

Risultati: Identificazione di "Trappole per turisti" (voti turisti > locali) o "Preferiti dai locali".

Casi d'uso: Consigli di viaggio autentici basati sulle preferenze dei locali.

3.8 8. Preferenze Stagionali

Obiettivo: Analizzare come varia il gradimento in base alla stagione e al tipo di viaggio (Leisure/Business).

Logica: Estrazione del mese dalla data recensione per determinare la stagione. Parsing dei tag per identificare il tipo di viaggio. Aggregazione combinata.

Tecnologie: Date functions, String matching su tags.

Risultati: Performance degli hotel in specifiche stagioni (es. hotel ottimi per l'estate ma carenti in inverno).

Casi d'uso: Pianificazione viaggi in base al periodo.

3.9 9. Analisi Durata Soggiorno

Obiettivo: Correlare la durata del soggiorno al livello di soddisfazione.

Logica: Utilizzo di Regular Expressions per estrarre il numero di notti dal campo **Tags** (es. "Stayed 3 nights"). Categorizzazione in Short, Medium, Long stay e calcolo media voti.

Tecnologie: `regex.extract`, conditional logic (`when/otherwise`).

Risultati: Statistiche che mostrano se i soggiorni lunghi tendono ad avere recensioni peggiori o migliori.

Casi d'uso: Ottimizzazione offerte per soggiorni lunghi/corti.

3.10 10. Esperienza del Recensore

Obiettivo: Valutare se i recensori esperti sono più critici dei novizi.

Logica: Segmentazione dei recensori in base al campo

`Total_Number_of_Reviews_Reviewer_Has_Given` (Novice, Intermediate, Expert).

Confronto delle distribuzioni dei voti.

Tecnologie: Bucketizer o logica condizionale personalizzata.

Risultati: Analisi della severità del voto in funzione dell'esperienza.

Casi d'uso: Ponderazione del peso delle recensioni in un sistema di ranking avanzato.

4 Conclusioni Finali

Il progetto ha dimostrato con successo come l'utilizzo di **Spark** permetta di effettuare analisi complesse e multidimensionali su un dataset di grandi dimensioni con tempi di risposta contenuti. L'architettura implementata garantisce scalabilità orizzontale, potendo gestire volumi di dati ben superiori a quello attuale senza modifiche al codice.

Obiettivi Soddisfatti: Tutti i requisiti di analisi descrittiva, diagnostica e predittiva (clustering/trend) sono stati implementati. L'integrazione con Streamlit rende i risultati accessibili e navigabili.

Possibili Sviluppi Futuri:

- **Analisi del Testo Avanzata:** Implementazione di modelli NLP (es. BERT) per Sentiment Analysis granulare sulle recensioni testuali, andando oltre il semplice voto numerico.
- **Streaming:** Integrazione con Spark Structured Streaming per elaborare recensioni in tempo reale.
- **Raccomandation System:** Sviluppo di un motore di raccomandazione collaborativo basato sulla similarità utente-utente trovata nel cluster analysis.