Relazione Progetto Big Data: Hotel Reviews Analytics

Giuseppe Pasquale Caligiure - Mat. 280867

2026

Indice

1 Presentazione del Progetto

1.1 Obiettivi Realizzati

Il progetto "Hotel Reviews Analytics" ha l'obiettivo di analizzare un dataset massivo di recensioni alberghiere per estrarre insight significativi utilizzando tecnologie Big Data. Attraverso l'elaborazione distribuita, sono stati realizzati diversi moduli di analisi che coprono aspetti temporali, testuali, geospaziali e comportamentali. 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 Tecnologie Utilizzate

Il progetto è stato sviluppato utilizzando uno stack tecnologico moderno orientato all'analisi dati su scala:

- **Linguaggio**: Python 3.11.
- **Core Computing**: **Apache Spark** (PySpark) per l'elaborazione parallela e distribuita dei dati. Utilizzo intensivo di DataFrame API, Spark SQL, Window Functions e User Defined Functions (UDF).
- **Machine Learning**: **Spark MLlib** per operazioni di featurizzazione e clustering (K-Means) e **Scikit-learn** per regressioni lineari specifiche all'interno di UDF pandas.
- **Frontend**: **Streamlit** per la creazione di una web application interattiva che permette all'utente di eseguire query e filtrare risultati.
- **Visualizzazione**: **Altair** e **PyDeck** per la creazione di grafici interattivi e mappe geospaziali.
- **Gestione Dipendenze Windows**: Winutils e Hadoop binaries per l'esecuzione locale su ambiente Windows.

1.3 Architettura Frontend/Backend

L'applicazione segue una logica semi-disaccoppiata:

- **Backend (Spark)**: Il file `queries.py` contiene la logica di business. Ogni funzione di analisi accetta un DataFrame Spark in input e restituisce un DataFrame Spark trasformato con i risultati. Questo livello gestisce la complessità dell'elaborazione distribuita.
- **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.

1.4 Logica di Funzionamento e Requisiti

L'utente avvia l'applicazione tramite script batch o comando Streamlit. Dopo il caricamento iniziale del dataset in memoria (DataFrame Spark), tramite una sidebar laterale è possibile selezionare una delle query disponibili. Ogni query espone parametri specifici (es. numero minimo di recensioni, raggio in km) modificabili tramite slider o input box. L'esecuzione avviene on-demand sfruttando il motore Spark.

2 Descrizione del Dataset Hotel_Reviews

Il dataset utilizzato è `Hotel_Reviews.csv`, un archivio contenente recensioni di hotel di lusso in Europa.

- **Dimensione File**: Circa 238 MB.
- **Numero di Righe**: Oltre 515.000 recensioni.

2.1 Descrizione dei Campi

- `Hotel_Address`: Indirizzo fisico dell'hotel.
- `Additional_Number_of_Scoring`: Metrica di scoring aggiuntiva.
- `Review_Date`: Data in cui è stata rilasciata la recensione.
- `Average_Score`: Punteggio medio storico dell'hotel.
- `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`: Storico recensioni dell'utente.
- `Reviewer_Score`: Voto assegnato dal recensore nella specifica istanza.
- `Tags`: Lista di stringhe che descrivono il soggiorno (es. "Leisure trip", "Couple", "Stayed 2 nights").
- `days_since_review`: Giorni trascorsi dallo scraping.
- `lat` / `lng`: Coordinate geografiche (Latitudine/Longitudine).

3 Descrizione delle Query Implementate

3.1 1. Top Hotel per Nazione

Obiettivo: Identificare le eccellenze alberghiere suddivise per paese.
Logica: La query estrae la nazione dall'indirizzo dell'hotel, raggruppa le strutture per nazione e le ordina in base al punteggio medio (`Average_Score`) e al numero di recensioni (come tie-breaker).
Tecnologie: Spark SQL Functions, Window Functions (per il ranking).
Risultati: Lista dei top N hotel per ogni nazione presente nel dataset (UK, France, Italy, etc.).
Casi d'uso: Utenti che cercano i migliori hotel assoluti in una specifica destinazione turistica.

3.2 2. Trend Temporale delle Recensioni

Obiettivo: Capire se un hotel sta migliorando o peggiorando nel tempo.
Logica: Le recensioni vengono raggruppate per hotel e ordinate cronologicamente. Viene applicata una regressione lineare (score vs tempo) per calcolare la pendenza (slope) del trend.
Tecnologie: Pandas UDF (User Defined Function) per parallelizzare l'esecuzione di `scikit-learn` `LinearRegression` su ogni gruppo di hotel distribuito nei nodi Spark.
Risultati: Coefficiente di trend per ogni hotel. Slope positivo indica miglioramento, negativo peggioramento.
Casi d'uso: Identificare "stelle nascenti" o hotel decadenti nonostante un alto punteggio medio storico.

3.3 3. Analisi Influenza Tag (MapReduce)

Obiettivo: Determinare quali fattori (es. "Single Room", "No Window") impattano positivamente o negativamente sul voto.
Logica: Segue il paradigma MapReduce. Fase Map: "esplosione" della stringa dei tag in righe singole. Fase Reduce: aggregazione per singolo tag calcolando la media dei voti delle recensioni in cui appare e lo scostamento dalla media globale.
Tecnologie: `explode`, `groupBy`, aggregazioni statistiche.
Risultati: Classifica dei tag con il maggiore impatto positivo e negativo (Impact Score) e relativo indice di affidabilità.
Casi d'uso: Gestori hotel che vogliono capire quali caratteristiche sono più apprezzate o criticate.

3.4 4. Local Competitiveness (Analisi Geospaziale)

Obiettivo: Confrontare le performance di un hotel rispetto ai suoi diretti concorrenti geografici.
Logica: Esegue un self-join del dataset basato sulla distanza geografica (Formula di Haversine). Ogni hotel viene confrontato con tutti gli altri hotel entro un raggio K km. Viene calcolato il delta tra il punteggio dell'hotel e la media del vicinato.
Tecnologie: Join cartesiano ottimizzato con filtri geospaziali, funzioni trigonometriche Spark.
Risultati: Identificazione di "Local Gems" (punteggio alto in zona mediocre) e "Underperformers".
Casi d'uso: Analisi di mercato competitiva per area geografica.

3.5 5. Segmentazione Hotel (K-Means Clustering)

Obiettivo: Raggruppare gli hotel in cluster omogenei basati su caratteristiche multidimensionali.
Logica: Vengono estratte feature come Punteggio, Popolarità (numero recensioni), Verbosità

delle recensioni e Bias di nazionalità. I dati vengono normalizzati e processati dall'algoritmo K-Means.
Tecnologie: `Spark MLlib` (VectorAssembler, StandardScaler, KMeans pipeline).
Risultati: Assegnazione di ogni hotel a un cluster (es. "Hotel Popolari di Lusso", "Hotel Economici di Nicchia").
Casi d'uso: Segmentazione di marketing, raccomandazioni di hotel simili.

3.6 6. Bias Nazionalità Recensore

Obiettivo: Capire se certe nazionalità tendono a dare voti più alti o bassi rispetto alla media.
Logica: Raggruppamento per `Reviewer_Nationality` e calcolo della media voti e deviazione standard.
Tecnologie: Aggregazioni Spark standard.
Risultati: Profilo di voto per nazione (es. "Gli utenti da X tendono a votare basso").
Casi d'uso: Normalizzazione dei voti in piattaforme internazionali.

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: `regexp_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.