MondoDBInventory

Dataset	2
Campi del dataset	2
Preprocessing	3
Preprocessing - Pandas	3
Preprocessing - Valori nulli	3
Slicing	4
Image	4
Category	4
Selling Price	4
About Product	4
Technical Details	4
Shipping Weight	4
Progettazione della WebApp	5
Scelta del Database NoSQL	5
Modello architetturale della Web-App	5
Struttura dell’applicazione	5
Back-end	5
ProductLoader.java	5
ProductQuery.java	5
ConPool.java	6
ProductDAO.java	6
ServletInit.Java	6
Indici	7

MongoDBInventory è una applicazione web nata per realizzare un inventario digitale che permette di aggiungere, cancellare e modificare prodotti Amazon e di visualizzare tutte le loro informazioni. I prodotti sono immagazzinati in un database NoSQL ovvero MongoDB in una collezione chiamata AmazonProducts. Lo scopo dello sviluppo di questa applicazione è stato principalmente quello di comprendere come utilizzare MongoDB e usare concetti l'utilizzo degli indici sui dati memorizzati.

Dataset

Il dataset utilizzato per questo progetto è "*Amazon Product Dataset 2020*" ottenuto tramite https://data.world .

Il dataset originale contiene circa 30'000 record, mentre la versione disponibile su "*data.world*" ne contiene. Durante la fase di preprocessing, per velocizzare la fase di progettazione e di costruzione del database ne sono stati impiegati 500, considerati i più rappresentativi.

Amazon Product Dataset 2020 è un database con estensione .csv ed è stato creato tra il 1 Gennaio 2020 e il 31 Gennaio 2020.

Campi del dataset

I campi del dataset sono:
- Uniq Id
- Product Name
- Brand Name
- Asin
- Category
- Upc Ean Code
- List Price
- Selling Price
- Quantity
- Model Number
- About Product
- Production Specification
- Technical Details
- Shipping Weight
- Product Dimensions
- Image
- Bariants
- SKU
- Product Url
- Stock
- Product Details
- Dimensions
- Color
- Ingredients
- Direction To Use
- Is Amazon Seller
- Size Quantity Variant
- Product Description

Soltanto alcuni di questi campi sono stati effettivamente mantenuti. Nel capitolo successivo esamiamo quali sono, insieme alle rispettive operazioni di preprocessing.

Preprocessing

Preprocessing - Pandas

Per il preprocessing è stato impiegato Python 3.8 e nello specifico la libreria *Pandas.*
Questa libreria permette di fare analisi e manipolazione dei dati ed è open source. Fornisce strumenti per memorizzare dati sottoforma di file con estensione come .json, .csv, etc.. nonchè strumenti per la memorizzazione in memoria principale sottoforma di datagrammi. I datagrammi rappresentano una vista virtuale e tabellare dei dati, attraverso i quali è possibile manipolare righe e colonne.

Preprocessing - Valori nulli

Per prima cosa viene affrontato il tema dei valori nulli. É stato dunque effettuato il drop attraverso il comando data.drop[‘column name’, axis=1] di tutti quei campi i quali non contenevano alcun valore.

I campi in questione sono:
- Brand Name
- Asin
- Upc Ean Code
- List Price
- Quantity
- Model Number
- Product Specification
- Product Dimensions
- Variants
- Sku
- Stock
- Product Details
- Dimensions
- Color
- Ingredients
- Direction To Use
- Is Amazon Seller
- Product Description
- Size Quantity Variant

In questo modo abbiamo ridotto i campi da considerare da a 28 a 8 campi, ovvero:
- **Product Name** : nome del prodotto.
- **Category** : categoria del prodotto.
- **Selling Price**: prezzo di vendita del prodotto.
- **About Product** : descrizione del prodotto.
- **Technical Details** : descrizione tecnica del prodotto.
- **Shipping Weight** : peso in kg del prodotto.
- **Image** : URI dove poter reperire l’immagine del prodotto.
- **Product Url**: URL della pagina Amazon dove poter effettuare l’acquisto del prodotto.

Una volta effettuato il drop delle colonne totalmente vuote, abbiamo deciso di affrontare il problema dei valori nulli presenti in campi parzialmente pieni. I campi interessati sono stati:
- Category
- About Product
- Technical Details
- Shipping Weight

Slicing

Data la grande mole del dataset di partenza contenente circa 30’000 record, per velocizzare e facilitare la fase di progettazione e di realizzazione abbiamo mantenuto 500 record sulla base dei campi che avevamo precedentemente estratto.

Image

Il campo *Image* del dataset comprendeva diversi URL relative ad immagini del prodotto da diverse angolazioni. Piuttosto che tenere traccia di tutte queste URL abbiamo preso in considerazione soltanto la prima URL relativa all’immagine più rappresentativa dell’oggetto.

Le URL erano contenute in un unica stringa separate dal carattere ‘|’. Quello che è stato fatto è splittare la stringa principale in sotto stringhe indicando il carattere ‘|’ come carattere di separazione ed infine prendere il primo elemento.

Category

Il campo Category aveva lo stesso problema. Questo campo assumeva valori lungo l’intera gerarchia del campo categoria. Piuttosto quindi che gestire granularità multiple abbiamo deciso di dotare i singoli prodotti della granularità più bassa in modo tale da avere un numero di categorie non dispersivo con cui trattare i dati.

Selling Price

È stato rimosso il simbolo del dollaro $, convertito il campo stringa in campo float ed infine convertito nella valuta Euro.

About Product

Il campo About Product racchiudeva la descrizione del prodotto attraverso una serie di frasi separate dal carattere ‘|’ quello che è stato fatto è dunque eliminare questo carattere di separazione e unire queste singole stringhe in un’ unica stringa.

Technical Details

Questo campo, come About Product soffriva del problema della presenza del carattere di separazione ‘|’. Quello che è stato fatto è dunque creare una singola stringa a partire dalle singole stringhe riportate in questo campo.

Shipping Weight

Questo campo era rappresentato in punds e ounce. É stata dapprima eliminata l’unità di misura del peso, per poi convertire la stringa risultante in float.
Una volta ottenuto il numero relativo al peso del prodotto, è stato convertito tutto in kg ed arrotondata la precisione della virgola a due cifre decimali.

Progettazione della WebApp

Scelta del Database NoSQL

La nostra scelta è ricaduta sull'utilizzo di **MongoDB**, il quale è un DBMS non relazionale orientato a documenti. I documenti che utilizza MongoDB hanno una estensione JSON più dinamica, ovvero l'estensione BSON che rende l'integrazione dei dati in alcune implementazioni più facile e veloce. Per facilitare l'inserimento e dunque la creazione del database a partire dal dataset preprocessato, è stato fatto uso anche di **MongoDB Compass** il quale attraverso una GUI permette di effettuare query, aggregare ed analizzare i dati in maniera visuale.

Il database è stato costruito su cloud sfruttando il servizio **MongoDB Atlas**. Il DB realizzato è stato chiamato **AmazonProducts** così come la collezione in cui sono contenuti i diversi prodotti.

Modello architetturale della Web-App

Il modello utilizzato per la creazione dell Web-App è il modello MVC. Esso rappresenta un pattern architetturale molto diffuso per la creazione di applicazioni web, e permette di separare la logia di presentazione dei dati dalla logica di business.

Questo modello si basa su tre componenti fondamentali:
- **Model**: il model fornisce i metodi per accedere ai dati utili all'applicazione e aggiorna di conseguenza l'interfaccia grafica sottoposta all'utente.
- **View**: rappresenta la dashboard sottoposta all'utente tramite il quale esso può inserire, cancellare e modificare gli elementi contenuti nel database graficamente.
- **Controller**: il controller riceve i comandi dell'utente che ha interagito precedentemente con una view e sfruttando il model riflette questi cambiamenti all'interno del database.

Struttura dell'applicazione

Back-end

Lato Back-end la struttura è data dalle seguenti 3 directory:
- **controller**
  - ProductLoader.java
  - ProductQuery.java
- **model**
  - ConPool.java
  - ProductDAO.java
- **utils**
  - ServletInit.java

ProductLoader.java

Classe java che permette il reperimento di tutti i dati dal DB sfruttando il metodo del ProductDAO getDataFromDB(). É la prima servlet che viene invocata quando il caricamento della pagina web è giunto al termine. In questo modo tutti i dati vengono reperiti e visualizzati.

ProductQuery.java

Permette di effettuare tutte le operazioni espresse nel ProductDAO.java. Le operazioni previste sono: inserimento, cancellazione, modifica e reperimento delle informazioni dei diversi prodotti.

Questa servlet conduce anche una serie di operazioni di filtraggio sui diversi campi del prodotto al momento in cui viene effettuata una query.

ConPool.java

È una semplice classe che permette di andare a stabilire la connessione con il DB. Viene impiegata ogni volta che andiamo ad effettuare una operazione sul database. Fortemente utilizzata da ProductDAO.java.

ProductDAO.java

Dispone dei metodi:

- public JSONObject **getDataFromDB** ()
  - Stabilisce una connessione con il Database, per poi reperirne tutti i dati memorizzati in un JSONArray che viene restituito.

- public JSONObject **getResultQuery** (String productName, String selectedValue, String minPrice, String maxPrice, String minWeight, String maxWeight)
  - Dopo aver stabilito una connessione don il Database effettua una query in base al nome del prodotto, il prezzo minimo/massimo, il peso minimo/massimo e la categoria. Restituisce un JSONArray contenente il risultato della query.

- public JSONObject **insertData** (String pName, String category, double price, double weight, String aboutProd, String techDet, String prodUrl, String imageUrl)
  - Crea una connessione con il database. Viene quindi creato un nuovo documento con i diversi campi del prodotto e inseriti all‘interno della collezione.

- public JSONObject **getProductByID**(String idProduct)
  - Crea una connessione con il database e reperisce un determinato documento in base all’ object id del prodotto.

- public JSONObject **deleteProduct** (String idProduct)
  - Crea una connessione con il database ed elimina un documento in base all’ object id del prodotto.

- public JSONObject **editData** (String idProduct, String pName, String category, double price, double weight, String aboutProd, String techDet, String prodUrl, String imageUrl)
  - Crea una connessione con il database e in base ai parametri dati in input andiamo a modificare il documento che ha per object id quello dato in input.

- public JSONObject **avgWeightAggregate** ()
  - Crea una connessione con il database e effettua un’ aggregazione sul campo ‘’weight”. Per ogni categoria viene quindi calcolato il peso medio.

- public JSONObject **avgPriceAggregate** ()
  - Crea una connessione con il database e efettua un’ aggregazione sul campo ‘’Price”. Per ogni categoria viene quindi calcolato il prezzo medio.

ServletInit.Java

Classe necessaria per l’inizializzazione della landing page rappresentata dalla jsp ‘’index.jsp”

Indici

- Gli indici sono stati creati sui campi che vengono acceduti più frequentemente ovvero:
- **Category**: è stato creato un indice di tipo single field. L'ordine adottato è quello crescente (1).
- **_Id**: è di tipo single field, definito da MongoDB in automatico. L'ordine adottato di default è quello crecente (1).
- **Selling Price, Shipping Weight**: è di tipo compound. Per Selling price è stata stabilita una regola di sort crescente (1) mentre per Shipping Weight è stata stabilita una regola di sort decrescente (-1)
- **Product Name**: ha la proprietà di essere un compound index testuale. Questo ci permette di effettuare una ricerca su stringhe all'interno della collezione.