Relazione elaborato Big Data 2019 / 2020
Gruppo A

Flavio Biancospino 59972
Andrea D'Acunto 61006
Vincenzo Russo 61005
Rocchina Valanzano
Cristian Zampino 60857

Indice

Introduzione

I dati analizzati in questo elaborato fanno parte del progetto del professor Julian McAuley [1] nell'ambito della realizzazione di un sistema di raccomandazione attraverso l'analisi delle recensioni di Amazon. Sulla falsariga del progetto citato, è stato realizzato un sistema di raccomandazione attraverso l'analisi delle recensioni dei prodotti Amazon appartenenti alla categoria *Sports & Outdoors*.
Nel seguente capitolo saranno illustrate le modalità utilizzate dal gruppo di progetto per l'accesso, l'analisi, la manipolazione e la visualizzazione dei dati.

Strumenti software

MongoDB

Considerata l'elevata dimensione dei dati - collezionati in due differenti file JSON - si è reso necessario l'utilizzo di un database non relazionale. Tra le varie possibilità, la scelta del gruppo di progetto è ricaduta su *MongoDB*, uno dei più noti database non relazionali (NoSQL). Si tratta di una soluzione orientata ai documenti che sfrutta il formato JSON per la memorizzazione e la rappresentazione dei dati.
Oltre a fornire una piattaforma di hosting per i dati, MongoDB offre alcuni peculiari strumenti software quali *Atlas*, *Compass* e *Charts* utili, rispettivamente, per la gestione dei dati via web, per la gestione dei dati sulla propria macchina locale e per la loro visualizzazione.

Jupyter Notebook

Per la scrittura e l'esecuzione del codice, il gruppo di progetto si è affidato alla soluzione offerta dai *Jupyter Notebook*. Questi forniscono un ambiente di lavoro integrato che consente di creare ed eseguire blocchi di codice. I linguaggi di programmazione supportati sono molteplici e tra questi figura il linguaggio Python, utilizzato massivamente nel corso della realizzazione del progetto.

Librerie Python

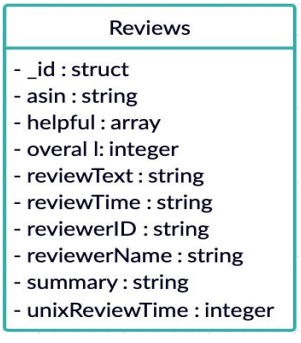
Il linguaggio utilizzato per la realizzazione del progetto è Python. Le librerie Python utilizzate per l'elaborazione e la manipolazione dei dati sono: *PySpark*, per l'interrogazione dei dati e la conseguente gestione dei calcoli paralleli associati; *scikit-learn*, per l'utilizzo di metodi per l'apprendimento automatico (e.g, clustering); *NLTK*, per le elaborazioni del linguaggio naturale. Invece, per la parte inerente la visualizzazione dei dati, sono state utilizzate le seguenti librerie: *Matplotlib*, *Plotly* e *WordCloud*.

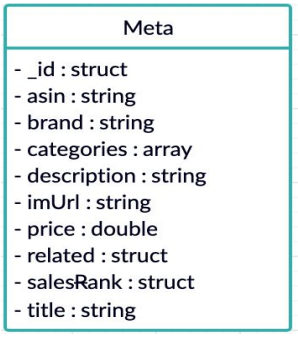
Tableau

Per la visualizzazione dei dati, un ulteriore strumento software utilizzato è *Tableau*, il quale fornisce funzionalità avanzate di preparazione e grafica dei dati unite a una grafica intuitiva e di facile fruizione.

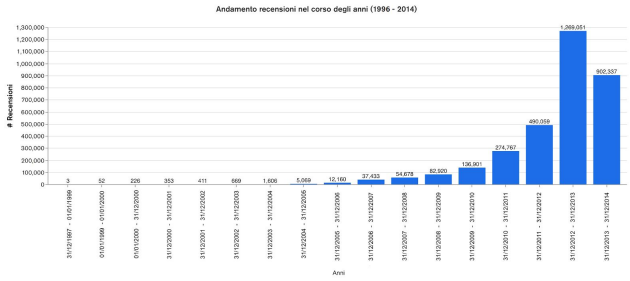
Analisi dei dati

I dati fornitici sono suddivisi in due collezioni JSON: *reviews* e *meta*. La prima collezione contiene i dati relativi alle recensioni dei prodotti; la seconda collezione contiene i dati relativi ai prodotti. Nelle due immagini sottostanti sono rappresentati gli schemi generati da MongoDB per le due collezioni JSON.

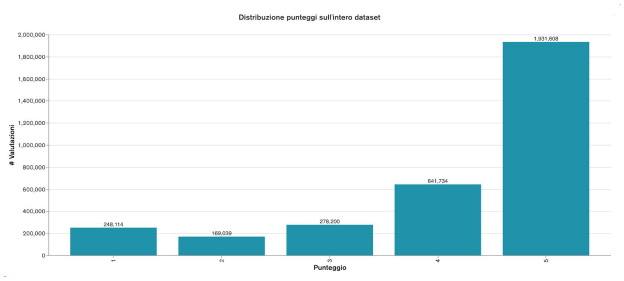




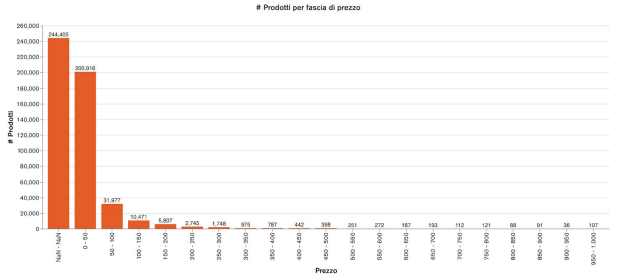
Attraverso lo strumento *Charts* offerto da MongoDB siamo stati in grado di rappresentare l'andamento di alcuni attributi da noi ritenuti significativi.

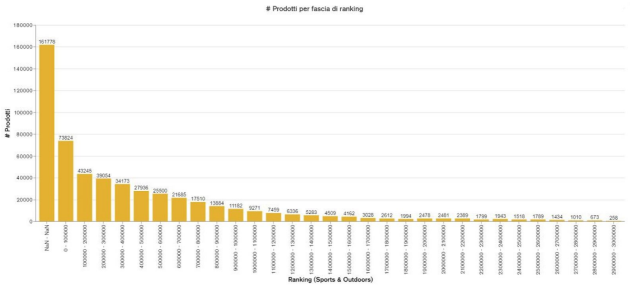


Dal primo grafico si evince che, a partire dall'anno 2006, c'è stato un incremento esponenziale del numero di recensioni. Questo è probabilmente dovuto all'espansione di internet nel mondo e, conseguentemente, all'adozione del servizio Amazon.



Dal secondo grafico si evince che la maggior parte degli utenti di Amazon ha scritto una recensione positiva sui prodotti.





Terzo e quarto grafico rappresentano, rispettivamente, la distribuzione dei prezzi e la distribuzione del ranking. Si può notare come, per il prezzo, la maggior parte dei prodotti sia nella fascia 0-50$ e che, per il ranking, la maggior parte dei prodotti sia nella fascia 0-100000.
Per una migliore visualizzazione dei grafici si può far riferimento ai file contenuti nella cartella di progetto figs.

Attraverso lo strumento *Compass* offerto da MongoDB siamo stati in grado di attuare una prima analisi sulla distribuzione dei dati e controllare la percentuale di valori nulli per collezione. La collezione reviews non contiene valori nulli al proprio interno; la collezione meta contiene invece molti valori nulli per colonna. Nel seguito è illustrata la percentuale di valori nulli per colonna nella collezione meta.

asin	brand	categories	description
0%	71.35%	0%	24.39%

imUrl	price	related	salesRank	title
0.11%	45.92%	27.07%	8.61%	0.43%

Tali percentuali sono di estremo interesse e saranno utili nella successiva fase di preparazione dei dati.

Svolgimento

Preparazione dei dati

Dall'analisi preliminare fatta con MongoDB sono emersi diversi valori nulli e NaN. Per questo, prima di cominciare a lavorare con i dati in nostro possesso, sono state effettuate delle operazioni di pulizia.

Reviews

Nella collezione reviews è presente l'attributo *helpful*, il quale rappresenta l'utilità della recensione. Da questo attributo, di tipo array, sono state estratte due informazioni, la percentuale di utilità della recensione e il numero di persone che hanno trovato utile la recensione.

```
# Create helpful_rate and helpful_pos columns from helpful column.
df_reviews = df_reviews.rdd \
  .map(lambda x: \
       x + (get_helpful_rate(x['helpful']), x['helpful'][0], )) \
  .toDF(df_reviews.columns + ['helpful_rate', 'helpful_pos'])   \
  .drop('helpful')
```

La funzione `get_helpful_rate` utilizzata nel blocco di codice precedente estrae dall'attributo *helpful* la percentuale di persone che ritengono la recensione utile.

Meta

Nella collezione meta l'attributo *salesRank* risulta essere una struttura complessa contenente al suo interno più attributi, di cui solo uno di interesse. L'estrazione del valore di interesse è avvenuta attraverso il codice presente nel seguito.

```
# Create sales_rank columns from salesRank column.
df_meta = df_meta \
  .join(df_meta.select(['asin', 'salesRank.Sports &amp;amp;Outdoors']), \
                       on='asin') \
  .drop('salesRank') \
  .withColumnRenamed('Spor&amp;amp; Outdoors', 'sales_rank')
```

Di scarso interesse per l'analisi dei dati sono invece gli attributi *categories, related e imUrl*, contenenti al loro interno, rispettivamente, la lista delle categorie a cui appartiene il prodotto, i prodotti correlati e l'URL dell'immagine del prodotto.

```
# Drop categories, related and imUrl columns.
df_meta = df_meta.drop('categories', 'related', 'imUrl')
```

Sulla collezione meta è stato effettuato un calcolo preliminare dei valori nulli attraverso lo strumento *Compass* offerto da MongoDB. Un'ulteriore conferma della presenza di valori nulli è data dal seguente blocco di codice, il quale calcola il numero esatto di valori nulli per attributo.

```
# Count null values per column.
df_meta.select([count(when(isnull(c), c)).alias(c) \
                for c in df_meta.columns])
```

L'esecuzione del codice restituisce la seguente tabella, che dimostra la necessità di agire nei confronti dei valori nulli.

asin	brand	description	price	title	sales_rank
0	379710	129826	244405	2296	161778

Per risolvere questo problema sono state eseguite le seguenti operazioni:

- per gli attributi numerici, si è attuata la tecnica di *mean imputation* [2], la quale consiste nella sostituzione dei valori nulli con la media degli altri valori non nulli dello stesso attributo;

```
# Fill price null values or NaNs with mean value of the price
# column.
df_meta = df_meta.na.fill({'price': \
  df_meta.dropna(subset=['price'])  \
                .agg(avg('price')).first()[0]})
# Fill sales_rank_sports_etc null values with mean value of the
sales_rank column.
df_meta = df_meta.na.fill({'sales_rank': \
   df_meta.dropna(subset=['sales_rank']) \
                .agg(avg('sales_rank')).first()[0]})
```

- per gli attributi di tipo stringa, i valori nulli sono stati sostituiti con stringhe caratteristiche a seconda dell'attributo (e.g, 'No Brand' per l'attributo *brand*, 'No Title' per l'attributo *title*, 'No Description' per l'attributo *description*).

L'attributo *brand* è stato successivamente trasformato in un dato di tipo categorico attraverso il seguente blocco di codice.

```
# Map brand's name to categorical.
df_meta = df_meta.rdd \
  .map(lambda x: x + (brand_categorical[x['brand']], )) \
  .toDF(df_meta.columns + ['brand_cat'])
```

La funzione `brand_categorical` utilizzata nel blocco di codice precedente associa opportunamente a ogni brand - che ricordiamo essere un attributo di tipo stringa - un valore discreto.

Query

In questo paragrafo verranno elencate ed estensivamente spiegate le query prefissate. Ogni query è stata rappresentata opportunamente attraverso il software Tableau. E' possibile consultare i grafici facendo riferimento al file pdf Queries contenuto nella cartella di progetto pdf.

I 100 prodotti con il maggior numero di recensioni

```
df_1 = df_reviews \
   .groupby('asin') \
   .count() \
   .withColumnRenamed('count', 'reviews_count_product') \
   .orderBy('reviews_count_product', ascending=False)
```

Per ottenere i 100 prodotti con il maggior numero di recensioni si è prima applicata la funzione di raggruppamento sull'attributo *asin* della collezione reviews. Successivamente, si è effettuato un conteggio sugli attributi raggruppati. Infine, si è rinominato l'attributo relativo al conteggio e applicato un

ordinamento decrescente sul suddetto attributo.

I 100 reviewer che hanno effettuato il maggior numero di recensioni

```
df_2 = df_reviews \
  .groupby('reviewerID') \
  .count() \
  .withColumnRenamed('count', 'reviews_count_reviewer') \
  .orderBy('reviews_count_reviewer', ascending=False)
```

Per ottenere i 100 reviewer che hanno effettuato il maggior numero di recensioni si è dapprima applicata la funzione di raggruppamento sull'attributo *reviewerID* della collezione reviews. Successivamente, si è effettuato un conteggio sugli attributi raggruppati. Infine, si è rinominato l'attributo relativo al conteggio e applicato un ordinamento decrescente sul suddetto attributo.

Le 50 marche i cui prodotti sono stati maggiormente recensiti

```
df_3 = df_reviews \
  .join(df_meta, on='asin') \
  .filter("brand != 'No Brand'") \
  .groupBy('brand') \
  .count() \
  .withColumnRenamed('count', 'reviews_count_brand') \
  .orderBy('reviews_count_brand', ascending=False)
```

Per ottenere le 50 marche i cui prodotti sono stati maggiormente recensiti si è, nella fase iniziale, effettuata una join sull'attributo *asin* tra le due collezioni reviews e meta. Si è poi filtrato il risultato della join escludendo tutti i prodotti senza brand. Sul risultato si è applicata la funzione di raggruppamento sull'attributo *brand* - che risulta essere un valore univoco - e applicata una funzione di conteggio sugli attributi raggruppati. Infine, si è rinominato l'attributo relativo al conteggio e applicato un ordinamento decrescente sul suddetto attributo.

Le 50 marche i cui prodotti hanno un prezzo medio maggiore

```
df_4 =  df_reviews \
  .join(df_meta, on='asin') \
  .select(['brand', 'price']) \
  .filter("brand != 'No Brand'") \
  .groupby('brand') \
  .mean() \
  .withColumnRenamed('avg(price)', 'price_mean') \
  .orderBy('price_mean', ascending=False)
```

Per ottenere le 50 marche i cui prodotti hanno un prezzo medio maggiore si è, nella fase iniziale, effettuata una join sull'attributo *asin* tra le due collezioni reviews e meta. Si è poi filtrato il risultato della join selezionando i soli attributi *brand* e *price* ed escludendo tutti i prodotti senza brand. Sul risultato si è applicata una funzione di raggruppamento sull'attributo *brand* e applicata la funzione di aggregazione media sugli attributi raggruppati. In conclusione, si è rinominato l'attributo ottenuto dall'aggregazione e applicato un ordinamento decrescente sul suddetto attributo.

I 100 prodotti con le migliori recensioni

```
df_5 = df_reviews \
  .select(['asin', 'overall']) \
  .groupby('asin') \
  .mean() \
  .withColumnRenamed('avg(overall)', 'overall_mean_product')

# Join to df_1 in order to obtain reviews_count_product column.
df_5 = df_5 \
  .join(df_1, on='asin') \
  .orderBy(['overall_mean_product', 'reviews_count_product'], \
           ascending=False)
```

Per ottenere i 100 prodotti con le migliori recensioni si sono selezionati gli attributi *asin* e *overall* dalla collezione reviews. Si è poi applicata la funzione di raggruppamento sull'attributo *asin* e applicata la funzione di aggregazione media sugli attributi raggruppati. La colonna degli attributi raggruppati è stata opportunamente rinominata. Successivamente, si è effettuata una join sull'attributo *asin* tra il risultato delle operazioni appena descritte e il risultato del calcolo dei prodotti con il maggior numero di recensioni. La tabella risultante, contenente tre colonne di attributi di cui due numeriche, è stata ordinata inizialmente sull'attributo rappresentante il punteggio medio e, successivamente, sul numero di recensioni complessive del prodotto. Entrambi gli ordinamenti sono stati effettuati in maniera decrescente.

Le 50 marche con le migliori recensioni

```
df_6 = df_reviews \
  .join(df_meta, on='asin') \
  .select(['brand', 'overall']) \
  .filter("brand != 'No Brand'") \
  .groupBy('brand') \
  .mean() \
```

```
  .withColumnRenamed('avg(overall)', 'overall_mean_brand')

# Join to df_3 in order to obtain reviews_count_brand column.
df_6 = df_6 \
  .join(df_3, on='brand') \
  .orderBy(['overall_mean_brand', 'reviews_count_brand'], \
           ascending=False)
```

Per ottenere le 50 marche con le migliori recensioni si è effettuata una join sull'attributo *asin* tra la collezione reviews e la collezione meta. Si sono selezionati gli attributi *brand* e *overall* e filtrati i risultati escludendo i prodotti senza *brand*. In seguito, si è effettuato un raggruppamento sull'attributo *brand* e applicata la funzione di aggregazione media. La colonna degli attributi raggruppati è stata opportunamente rinominata. Successivamente, si è effettuata una join sull'attributo *brand* tra il risultato delle operazioni appena descritte e il risultato del calcolo delle marche i cui prodotti sono stati maggiormente recensiti. La tabella risultante, contenente gli attributi *brand, overall_mean_brand* e *reviews_count_product*, è stata ordinata in maniera decrescente su *overall_mean_brand* e *reviews_count_product*.

I 100 reviewer che hanno effettuato recensioni con la maggiore utilità media

```
df_mean = df_reviews \
  .select(['reviewerID', 'helpful_rate']) \
  .groupBy('reviewerID') \
  .mean('helpful_rate') \
  .withColumnRenamed('avg(helpful_rate)', 'helpful_rate_mean')

df_sum = df_reviews \
  .select(['reviewerID', 'helpful_pos']) \
  .filter('helpful_pos != 0') \
  .groupBy('reviewerID') \
  .sum() \
  .withColumnRenamed('sum(helpful_pos)', 'helpful_pos_sum')

df_7 = df_mean \
  .join(df_sum, on='reviewerID') \
  .orderBy(['helpful_rate_mean', 'helpful_pos_sum'], \
           ascending=False)
```

Per ottenere i 100 reviewer che hanno effettuato recensioni con la maggiore utilità media sono stati selezionati dalla collezione reviews gli attributi *reviewerID* e *helpful_rate*. Successivamente si è effettuato un raggruppamento sull'attributo *reviewerID* e applicata una funzione di aggregazione media sugli attributi raggruppati. La colonna degli attributi raggruppati è stata opportunamente rinominata. In maniera analoga, si è calcolato il numero di recensioni utili per *reviewerID*, con l'unica differenza che la funzione di aggregazione utilizzata è stata quella di somma. Si sono infine uniti i due risultati, sul calcolo dell'utilità media e sul calcolo della somma di voti utili per recensore, con una join sull'attributo *reviewerID*. Il tutto è stato ordinato in maniera decrescente.

I 100 reviewer che hanno effettuato recensioni con la minore utilità media

```
df_8 = df_7 \
  .orderBy(['helpful_rate_mean', 'helpful_pos_sum'], ascending=True)
```

Per ottenere i 100 reviewer che hanno effettuato recensioni con la minore utilità media è bastato ordinare il risultato della query precedente in maniera ascendente rimanendo invariate le priorità di ordinamento.

I 100 prodotti con il migliore ranking nelle vendite

```
df_9 = df_meta \
  .orderBy('sales_rank', ascending=True) \
  .select(['asin', 'sales_rank'])
```

Per ottenere i 100 prodotti con il migliore ranking nelle vendite si è effettuato un ordinamento ascendente sull'attributo *sales_rank* - minore è il rango del prodotto più in alto è la sua posizione in classifica - e selezionati solo gli attributi *asin* e *sales_rank*.

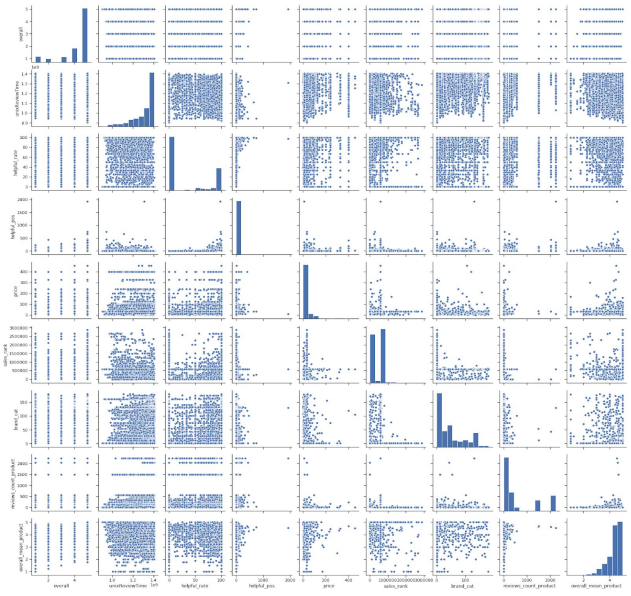
Le 50 marche i cui prodotti hanno il ranking medio migliore

```
df_10 = df_meta \
  .select(['brand', 'sales_rank']) \
  .filter("brand != 'No Brand'") \
  .groupby('brand') \
  .mean() \
  .withColumnRenamed('avg(sales_rank)', 'sales_rank_mean') \
  .orderBy(['sales_rank_mean'], ascending=True)
```

Per ottenere le 50 marche i cui prodotti hanno il ranking medio migliore si sono selezionati dalla collezione meta gli attributi *brand* e *sales_rank* ed esclusi gli elementi senza brand. Successivamente, si è effettuato un raggruppamento sul attributo *brand* ed applicata una funzione di aggregazione media sugli attributi raggruppati. La colonna degli attributi raggruppati è stata opportunamente rinominata. Infine, è stato applicato un ordinamento ascendente sul ranking medio.

Correlazioni

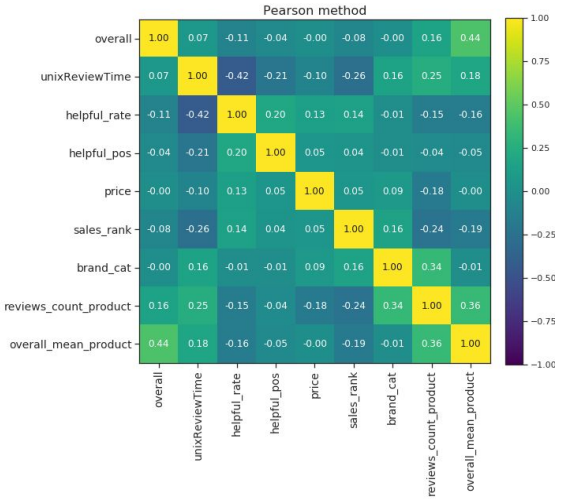
Al fine di calcolare le correlazioni si è reso necessario eliminare dalle due collezioni reviews e meta i valori non numerici. Successivamente, si sono unite queste due collezioni con i risultati delle operazioni per il calcolo dei prodotti con il maggior numero di recensioni e i prodotti con le migliori recensioni.
Ottenuto un DataFrame completamente numerico, la prima operazione eseguita è stata quella di graficare uno scatter plot per ogni combinazione di attributi (valutati binariamente). Questo passaggio è di estrema importanza perché permette di visualizzare l'eventuale presenza di correlazioni.



Per una migliore visualizzazione è possibile far riferimento al file png pairplot contenuta nella cartella di progetto figs.

Pearson

Visivamente, non sembra esistano **correlazioni lineari** evidenti. Per una verifica più accurata si è reso necessario calcolare la matrice di correlazione, che restituisce i coefficienti di correlazione di Pearson per ogni possibile combinazione binaria di attributi.



In riferimento alle richieste di progetto, si hanno i seguenti coefficienti di correlazione:

1. Prezzo di un prodotto - Punteggio medio ottenuto nelle recensioni: **0**;
2. Marca di un prodotto - Punteggio medio ottenuto nelle recensioni: **-0.01**;
3. Utilità di una recensione - Punteggio assegnato dalla recensione al prodotto: **-0.11**;
4. Data di una recensione - Utilità di una recensione: **-0.42**;
5. Data di una recensione - Punteggio assegnato al prodotto: **0.07**;
6. Ranking delle vendite - Punteggio ottenuto nella recensione: **-0.08**;
7. Numero delle recensioni di un prodotto - Ranking nelle vendite di un prodotto: **-0.24**;
8. Ranking nelle vendite di un prodotto - Prezzo di un prodotto: **0.05**.

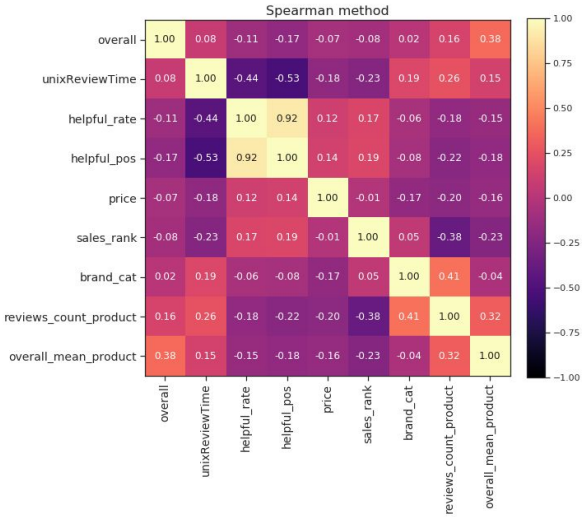
Grazie alla matrice di correlazione è possibile notare che esistono ulteriori correlazioni. Le più rilevanti sono le seguenti:

- Ranking nelle vendite di un prodotto - Data di una recensione: **-0.26**;
- Punteggio medio ottenuto nelle recensioni - Numero di recensioni del prodotto: **0.36**;
- Numero di recensioni di un prodotto - Brand del prodotto: **0.34**;
- Numero di recensioni di un prodotto - Data di una recensione: **0.25**;
- Brand del prodotto - Data della recensione: **0.16**;
- Punteggio medio del prodotto - Data di una recensione: **0.18**;

- Numero di recensioni di un prodotto - Prezzo di un prodotto: **-0.18**;
- Punteggio medio di un prodotto - Ranking nelle vendite: **-0.19**.

Spearman

Calcolati i coefficienti di correlazione lineari, è stato fatto un tentativo nel calcolo degli stessi attraverso il metodo di Spearman.
La differenza tra il metodo di Pearson e quello di Spearman è che il primo si limita a individuare le correlazioni lineari, il secondo individua quelle **correlazioni descritte da una generica funzione monotona** (crescente o decrescente).



In riferimento alle richieste di progetto, si hanno i seguenti coefficienti di correlazione:

1. Prezzo di un prodotto - Punteggio medio ottenuto nelle recensioni: **-0.16**;
2. Marca di un prodotto - Punteggio medio ottenuto nelle recensioni: **-0.04**;
3. Utilità di una recensione - Punteggio assegnato dalla recensione al prodotto: **-0.11**;
4. Data di una recensione - Utilità di una recensione: **-0.44**;
5. Data di una recensione - Punteggio assegnato al prodotto: **0.08**;
6. Ranking delle vendite - Punteggio ottenuto nella recensione: **-0.08**;
7. Numero delle recensioni di un prodotto - Ranking nelle vendite di un prodotto: **-0.38**;
8. Ranking nelle vendite di un prodotto - Prezzo di un prodotto: **-0.01**.

Ulteriori valori rilevanti sono:

- Numero di persone che hanno ritenuto utile la recensione - Utilità della recensione: **0.92**;
- Numero di persone che hanno ritenuto utile la recensione - Data di una recensione: **-0.53**;
- Utilità della recensione - Data di una recensione: **-0.44**;
- Numero di recensioni di un prodotto - Brand del prodotto: **0.41**;
- Numero di recensioni di un prodotto - Ranking di un prodotto: **-0.38**;
- Punteggio medio ottenuto nelle recensioni - Numero di recensioni di un prodotto: **0.32**;
- Ranking di un prodotto - Data di una recensione: **-0.23**;
- Numero di recensioni di un prodotto - Data di una recensione: **0.26**;
- Punteggio medio di un prodotto - Ranking di un prodotto: **-0.23**;
- Numero di recensioni di un prodotto - Prezzo di un prodotto: **-0.2**.

Come si può notare, alcuni coefficienti di correlazione differiscono da Pearson a Spearman. In generale: quando i due coefficienti non differiscono e sono prossimi allo zero, allora i due attributi sono indipendenti; quando i due coefficienti differiscono notevolmente l'uno dall'altro allora la relazione tra i due attributi è ben descritta da una funzione monotona non lineare (e.g., esponenziale con argomento negativo e iperbole).
Per una migliore visualizzazione delle matrici è possibile far riferimento al file png correlation contenuto nella cartella di progetto figs.

Analisi e raccomandazione

L'obiettivo del progetto è quello di creare un sistema di raccomandazione che ci permetta di individuare un prodotto con le seguenti caratteristiche:

- molte buone recensioni;
- poche cattive recensioni;
- molte recensioni in assoluto;
- trarre il maggior profitto possibile.

Analisi iniziale e preparazione

Il primo passo compiuto è stato quello di effettuare una scrematura sui dati in nostro possesso. Per far questo, sono stati presi in considerazioni i primi 100 prodotti con il maggior numero di recensioni. Questo è stato fatto, da un lato, per semplificare la ricerca del prodotto desiderato; dall'altro lato, per rendere le successive operazioni di ricerca del prodotto coerenti almeno con uno degli obiettivi posti - la ricerca di un prodotto con un numero elevato di recensioni in assoluto.

```
# Take the first 100 objects with most reviews.
df_most_reviews = df_meta.join(df_1.limit(100), on='asin')
```

Il passo successivo è stato quello di contare per ogni prodotto il numero di recensioni positive e negative. Una recensione viene definita negativa se il suo punteggio è inferiore a 3; una recensione viene definita positiva se il suo punteggio è maggiore di 3. In questa analisi sono state deliberatamente ignorate le recensioni con punteggio 3, considerate neutre.

```
# Count the number of good reviews and bad reviews per object.
# N.B. A review is considered good if its overall is greater
# than 3; bad if its overall is lesser than 3; neutral otherwise.
# Neutral reviews are deliberately ignored in this analysis.
df_good_reviews = df_reviews \
    .select('asin', 'overall') \
    .filter('overall > 3') \
    .groupBy('asin') \
    .count() \
    .withColumnRenamed('count', 'good_reviews')

df_bad_reviews = df_reviews \
    .select('asin', 'overall') \
    .filter('overall < 3') \
    .groupBy('asin') \
    .count() \
    .withColumnRenamed('count', 'bad_reviews')
```

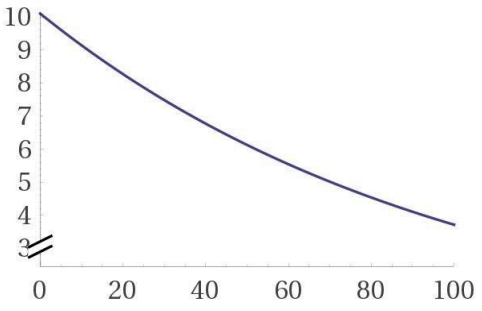
A partire dagli attributi *price* e *sales_rank*, è stata creata un'ulteriore feature chiamata *revenue* e identificante il guadagno sul singolo prodotto. Questo attributo è descritto da una funzione non lineare monotona decrescente - l'esponenziale con argomento negativo - che penalizza il singolo guadagno tanto più è alto il ranking del prodotto. Si ha che

$$revenue = price \cdot e^{\frac{1-rank}{k}},$$

dove:

- $revenue$ è il guadagno sul singolo prodotto;
- $price$ è il prezzo del singolo prodotto;
- $rank$ è il ranking del prodotto;
- k è un coefficiente di smorzamento, utile per controllare la decrescita della funzione esponenziale.

La definizione della precedente equazione è stata frutto di ragionamenti empirici. Nel seguito è rappresentata la funzione in questione con $price = 10$ e $k = 100$. Eventualmente, si può incrementare il parametro k per rendere il processo di penalizzazione di $price$ più lento.



Nel seguito è presentato il blocco di codice che permette di creare l'attributo *revenue* a partire da *price* e *sales_rank*.

```
# Create column <revenue>, that aggregate <price> and <sales_rank>
# columns with the following formula: <revenue> = <price> *
# exp((1 - <rank>) / mean_rank).
mean_rank = df_most_reviews.agg({'sales_rank': 'avg'}).first()[0]
df_most_reviews = df_most_reviews.rdd \
  .map(lambda x: \
       x + (x['price'] * math.exp((1 - x['sales_rank']) \
       / mean_rank), )) \
  .toDF(df_most_reviews.columns + ['revenue'])
```

Il coefficiente di smorzamento - k nell'equazione precedente - del blocco di codice è definito come il valor medio calcolato sulla colonna *sales_rank*.
A completare l'analisi e preparazione dei dati, è stata effettuata un'operazione di join sull'attributo *asin* tra i 100 prodotti maggiormente recensiti, di cui vengono selezionati solo gli attributi *asin* e *revenue*, il DataFrame contenente le informazioni relative al numero di buone recensioni per prodotto e il DataFrame contenente le informazioni relative al numero di cattive recensioni per prodotto.

```
df_most_reviews = df_most_reviews.select('asin', 'revenue')

df_result = df_most_reviews \
  .join(df_good_reviews, on='asin') \
  .join(df_bad_reviews , on='asin')
```

KMeans

Il passo successivo è stato quello di effettuare un'operazione di clustering utilizzando il KMeans sugli attributi *revenue*, *good_reviews* e *bad_reviews*.

```
X = np.asarray(X).reshape(df_result.count(), 4)
# features = [<revenue>, <good_reviews>, <bad_reviews>]
features = X[:, 1:].astype(np.float)
```

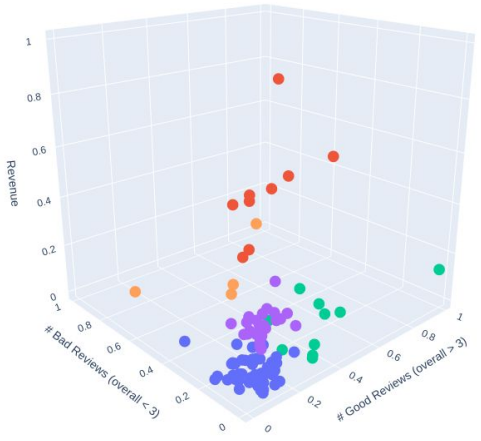
Prima di effettuare l'analisi con il KMeans, è stata effettuata un'operazione di features normalization utilizzando la tecnica del *Min-Max Scaler* [3] per normalizzare i valori nell'intervallo $[0; 1]$.

```
# Normalize features in range [0; 1].
scaler = MinMaxScaler()
scaler.fit(features)
features_norm = scaler.transform(features)
```

L'operazione di clustering è stata effettuata con la seguente riga di codice.

```
# N.B. KMeans results are not reproducible unless the keyword
random_state is not defined.
kmeans = cluster.KMeans(5, random_state=10).fit(features_norm)
```

Il risultato di questa operazione di clustering è rappresentato nell'immagine seguente.



Per una migliore visualizzazione dell'immagine si può far riferimento al file html 3d-scatter contenuto nella cartella di progetto figs.
Partendo dal grafico precedente, si è scelto di prendere in considerazione gli elementi di colore rosso, che identificano quei prodotti, tra i presenti, con il maggior profitto, un basso numero di recensioni negative e un buon numero di recensioni positive.

Analisi del testo

Filtrati quegli elementi appartenenti al solo cluster di colore rosso, il passo successivo è stato quello di effettuare un'analisi dei titoli e delle descrizioni dei prodotti filtrati. A questo scopo è stata utilizzata la libreria *NLTK*, per l'elaborazione del linguaggio naturale. Inizialmente, sono stati uniti tutti i testi dei titoli e delle descrizioni dei prodotti filtrati.

```
# Filter for cluster color.
asins_filtered = [X[i, 0] \
  for i in range(len(colors)) if colors[i] == '#EF553B']

titles_descriptions_filtered = df_meta \
  .select('asin', 'title', 'description') \
  .rdd.map(lambda x: [x['title'], x['description']] if x['asin'] in \
           asins_filtered else []) \
  .collect()

titles_descriptions_filtered = np.asarray( \
  [' '.join(a) for a in titles_descriptions_filtered if len(a) != 0])
text = ' '.join(titles_descriptions_filtered)
```

Successivamente, il testo risultante è stato ripulito di tutti i tag HTML e XML presenti ed è stata rimossa l'eventuale punteggiatura.

```
# Remove HTML or XML tags.
text_cleaned = re.sub('<[^<]+?>', '', text)

# Remove punctuation.
text_no_punctuation = text_cleaned \
  .translate(str.maketrans(' ', ' ', string.punctuation))
```

Il testo è stato successivamente diviso in token. Da questi token sono state escluse le parole comuni della lingua inglese e su ognuna realizzata un'analisi semantica, classificando i singoli token in nomi, verbi, pronomi, aggettivi, etc.

```
# Split text.
text_splitted = text_no_punctuation.split()

# Remove stopwords.
text_no_stopwords = [word for word in text_splitted]

# Identify splitted words as nouns, verbs, etc.
text_pos_tag = nltk.pos_tag(text_no_stopwords)
```

Sono stati presi in considerazione solo quei token classificati come nomi comuni singolari e plurali, nomi propri singolari e plurali.

```
text_filtered = []
for pair in text_pos_tag:
    word = pair[0]
    tag = pair[1]

    if tag in ['NN', 'NNS', 'NNP', 'NNPS']:
        text_filtered.append(word)
```

A partire dal testo filtrato, è stata rappresentata, attraverso la libreria Python *WordCloud*, la distribuzione di frequenza delle parole.

Chiaramente, l'immagine generata non identifica univocamente il prodotto. Tuttavia, è molto utile per comprendere quali sono le parole ricorrenti nei titoli e nelle descrizioni dei prodotti e quindi restringere la gamma delle possibili scelte. Nel seguito, verrà discusso come è stato individuato il prodotto.

Conclusione

Partendo dai dati appartenenti al cluster preso in considerazione (il cluster di colore rosso nella pagina precedente), si è scelto di attuare un'analisi grafica attraverso il software Tableau. In quest'analisi sono stati rapportati i diversi prodotti filtrati attraverso il clustering con gli obiettivi posti - numero di recensioni in assoluto del prodotto, numero di buone recensioni, numero di cattive recensioni e profitto.
Per questioni di spazio e per garantire una migliore fruizione del contenuto, i risultati delle analisi grafiche sono disponibili all'interno del file pdf presentation_graphs contenuto nella cartella di progetto pdf.
Il prodotto da noi individuato è un coltellino svizzero simil-Victorinox. Incidentalmente, questa scelta si trova in accordo con più parole ricorrenti nella distribuzione di frequenza (e.g, la parola *knife*, *blade* e *Victorinox*).
Analizzando i grafici di Tableau e focalizzando la nostra attenzione sul prodotto *Victorinox Swiss Army Classic SD Pocket Knife*, si può notare che questi ha:

- il secondo numero più alto di recensioni in assoluto, 1503;
- il numero più alto di recensioni positive, 1393;
- il numero - tra i più bassi - di recensioni negative, 63;
- un profitto medio che ammonta a 14,4$.

Individuato il prodotto, abbiamo successivamente cercato tutte quelle caratteristiche oggettive appartenenti al prodotto scelto. Per far questo, abbiamo analizzato tutti i titoli e le descrizioni del prodotto.
Il passo successivo è stato analizzare le recensioni negative del prodotto - con almeno una persona che ha ritenuto utile la recensione - per cercare quelle caratteristiche mancanti del prodotto o esistenti ma che allo stato attuale non soddisfano il cliente.
Alcune delle lamentele più comuni sono:

- la difficoltà nell'aprire gli utensili contenuti nel coltellino;
- la facilità con cui l'involucro di plastica esterno si graffia.

Riteniamo quindi che un prodotto di successo nella categoria *Sports & Outdoors* possa essere un coltellino svizzero con le seguenti caratteristiche:

- deve essere compatto;
- deve contenere al suo interno utensili quali le lame, il cacciavite, la pinzetta, lo stuzzicadente e le forbici;
- deve avere un sistema di apertura a bottone - un bottone per ogni utensile - in grado di permetterne la facile apertura;
- deve avere un involucro esterno di un materiale economico e antigraffio quale può essere una plastica adeguatamente verniciata o un materiale più naturale come il legno.

Riteniamo inoltre che il prezzo del prodotto debba aggirarsi sui 15$ - in linea con la concorrenza - per permettere un'adeguata mole di vendite.