



MODELLI E TECNICHE PER BIG DATA

Realizzazione di un'applicazione per l'analisi del carrello della spesa su un e-store

A.A 2022/23

Mario Saccomanno, mat.248124

Francesco Zumpano, mat.248188

9 febbraio 2023

Indice

1	Analisi del Dataset	2
1.1	Informazioni sui Prodotti	2
1.2	Informazioni sugli ordini	4
2	Sviluppo e funzionalità back-end	6
2.1	Descrizione Query	6
3	Sviluppo del front-end ed estrazione della conoscenza	10
3.1	Prodotti più acquistati	10
3.2	Statistiche temporali sugli acquisti	11
3.3	Numero oggetti in ordine	12
3.4	Prodotti venduti per reparto	12
3.5	Regole di associazione dei prodotti	13
3.6	Prodotti comprati insieme	13

Introduzione

Questo progetto ha avuto come obiettivo quello di realizzare un applicativo per effettuare query aggregate su un dataset riguardante una serie di transazioni effettuate sul sito InstaCart. Esso è un servizio di online grocery shopping, che permette ai suoi utenti di ordinare la propria spesa online ed averla consegnata direttamente al proprio domicilio. L'obiettivo delle query e dei relativi servizi esposti dal nostro applicativo è quello di effettuare una analisi esplorativa del dataset, per ottenere informazioni utili riguardo gli utenti di InstaCart e le loro abitudini. Si è voluta, dunque, effettuare quella che in gergo è detta **Market Basket Analysis**, o analisi del carrello della spesa. Essa è una tecnica usata dai grandi rivenditori volta a scoprire associazioni tra i prodotti venduti. Al cuore di questa tecnica vi è la ricerca dei gruppi di prodotti comprati frequentemente insieme, che possono rivelare informazioni interessanti riguardanti il comportamento degli utenti che effettuano gli acquisti. Questo applicativo ha, dunque, come obiettivo quello di fornire una piattaforma web per visualizzare ed esplorare questo set di dati.

Le tecnologie adottate nella realizzazione del progetto sono le seguenti:

- **Flutter** per la realizzazione delle pagine web
- **Cask** per l'esposizione di servizi REST
- **Apache Spark e il linguaggio Scala** per l'elaborazione del Dataset

Il documento è strutturato in 3 sezioni: la prima sezione si occuperà di descrivere il dataset su cui sono state effettuate la ricerca, analizzando nel dettaglio la struttura di ogni tabella contenuta in esso. Nella seconda sezione, invece, si parlerà della realizzazione del lato back-end. Mentre la terza sezione si occuperà dell'esposizione della parte visuale del nostro applicativo.

1 Analisi del Dataset

Come anticipato nell'introduzione, questo dataset è un insieme relazionale di file che descrivono più di 3 milioni ordini effettuati da oltre 200 mila utenti sul sito InstaCart, presentatisi a noi come file csv. Andiamo ad analizzare le tabelle dividendole in due gruppi: il primo è quello delle tabelle che danno informazioni sui prodotti venduti, mentre il secondo è quello delle tabelle che forniscono conoscenza sugli ordini effettuati.

1.1 Informazioni sui Prodotti

All'interno del dataset sono presenti 3 tabelle che offrono informazioni sui prodotti venduti su InstaCart e sulla classificazione interna al sito di questi ultimi. In particolare, la tabella **prodotti** fornisce informazioni riguardanti un singolo prodotto venduto su InstaCart. Essa è formata da quasi 50 mila righe, ognuna avente un id univoco, il nome del prodotto in questione, e le chiavi esterne afferenti al reparto ed al corridoio a cui esso appartiene.

```
root
|-- product_id: integer (nullable = true)
|-- product_name: string (nullable = true)
|-- aisle_id: string (nullable = true)
|-- department_id: string (nullable = true)
```

Figura 1: Schema di dati della tabella Products

product_id	product_name	aisle_id	department_id
1	Chocolate Sandwich...	61	19
2	All-Seasons Salt	104	13
3	Robust Golden Uns...	94	7
4	Smart Ones Classi...	38	1
5	Green Chile Anyti...	5	13
6	Dry Nose Oil	11	11
7	Pure Coconut Wate...	98	7
8	Cut Russet Potato...	116	1
9	Light Strawberry ...	120	16
10	Sparkling Orange ...	115	7
11	Peach Mango Juice	31	7
12	Chocolate Fudge L...	119	1
13	Saline Nasal Mist	11	11
14	Fresh Scent Dishw...	74	17
15	Overnight Diapers...	56	18
16	Mint Chocolate FL...	103	19
17	Rendered Duck Fat	35	12
18	Pizza for One Sup...	79	1
19	Gluten Free Quino...	63	9
20	Pomegranate Cranb...	98	7

Figura 2: Prime 20 righe della tabella Products

Le tabelle aisles.csv e departments.csv, rispettivamente, forniscono informazioni riguardanti i "corridoi" e reparti interni ad InstaCart con i quali vengono classificati e suddivisi i prodotti.

```
root
|-- aisle_id: integer (nullable = true)
|-- aisle: string (nullable = true)
```

Figura 3: Schema di dati della tabella Aisle

aisle_id	aisle
1	prepared soups sa...
2	specialty cheeses
3	energy granola bars
4	instant foods
5	marinades meat pr...
6	other
7	packaged meat
8	bakery desserts
9	pasta sauce
10	kitchen supplies
11	cold flu allergy
12	fresh pasta
13	prepared meals
14	tofu meat alterna...
15	packaged seafood
16	fresh herbs
17	baking ingredients
18	bulk dried fruits...
19	oils vinegars
20	oral hygiene

Figura 4: Prime 20 righe della tabella Aisle

```
root
|-- department_id: integer (nullable = true)
|-- department: string (nullable = true)
```

Figura 5: Schema di dati della tabella Departments

department_id	department
1	frozen
2	other
3	bakery
4	produce
5	alcohol
6	international
7	beverages
8	pets
9	dry goods pasta
10	bulk
11	personal care
12	meat seafood
13	pantry
14	breakfast
15	canned goods
16	dairy eggs
17	household
18	babies
19	snacks
20	deli

Figura 6: Prime 20 righe della tabella Departments

1.2 Informazioni sugli ordini

Le restanti 3 tabelle del dataset sono quelle che forniscono informazioni riguardanti gli ordini effettuati su InstaCart. Nella fattispecie l'autore del dataset ha deciso di dividere le informazioni riguardo il contenuto dell'ordine dalle informazioni non riguardanti il contenuto, generando così due tabelle. La tabella riguardante i contenuti dell'ordine, è stata a sua volta partizionata in due: *order_products_train.csv* e *order_products_prior.csv*, per fornire un set di dati da offrire in input ad un modello di machine learning in un momento successivo. Analizziamo la struttura della tabella *orders.csv* più in dettaglio. Essa è quella contenente le informazioni non contenutistiche dell'ordine, quali: identificativo dell'utente che ha effettuato quell'ordine, ora in cui è stato effettuato l'ordine, giorno della settimana in cui è stato effettuato l'ordine, giorni passati dal precedente ed una numerazione degli ordini effettuati da uno stesso utente (*order_number*).

```
root
|-- order_id: integer (nullable = true)
|-- user_id: integer (nullable = true)
|-- eval_set: string (nullable = true)
|-- order_number: integer (nullable = true)
|-- order_dow: integer (nullable = true)
|-- order_hour_of_day: integer (nullable = true)
|-- days_since_prior_order: double (nullable = true)
```

Figura 7: Schema di dati della tabella Orders

order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
2539329	1	prior	1	2	8	null
2398795	1	prior	2	3	7	15.0
473747	1	prior	3	3	12	21.0
2254736	1	prior	4	4	7	29.0
431534	1	prior	5	4	15	28.0
3367565	1	prior	6	2	7	19.0
550135	1	prior	7	1	9	20.0
3108588	1	prior	8	1	14	14.0
2295261	1	prior	9	1	16	0.0
2550362	1	prior	10	4	8	30.0
1187899	1	train	11	4	8	14.0
2168274	2	prior	1	2	11	null
1501582	2	prior	2	5	10	10.0
1981567	2	prior	3	1	10	3.0
738281	2	prior	4	2	10	8.0
1673511	2	prior	5	3	11	8.0
1199898	2	prior	6	2	9	13.0
3194192	2	prior	7	2	12	14.0
788338	2	prior	8	1	15	27.0
1718559	2	prior	9	2	9	8.0

Figura 8: Prime 20 righe della tabella Orders

Chiaramente il campo *days_since_prior_order* risulta essere *null* nel caso in cui quello

sia il primo ordine effettuato sulla piattaforma dall'utente. Procediamo ora all'analisi delle altre due tabelle contenenti informazioni sugli ordini, ovvero quelle relative ai prodotti comprati. Come menzionato in precedenza esse sono state partizionate in due tabelle, una di dimensione ridotta da usare come training set per un algoritmo di Machine Learning, ed un'altra contenente tutti i restanti ordini. Tale divisione implica due cose: la prima è che nella fase di elaborazione dei dati occorrerà effettuare l'unione di queste due tabelle, e la seconda è che esse avranno il medesimo schema di dati. Nello specifico ogni record delle tabelle *order_products* mantengono una chiave esterna al prodotto acquistato, un numero che rappresenta l'ordine con cui quel prodotto è stato aggiunto al carrello (se 2 vorrà dire che quel prodotto è stato il secondo messo nel carrello durante l'acquisto) ed un flag che segnala se quel prodotto era stato precedentemente acquistato da quell'utente.

```
root
|-- order_id: string (nullable = true)
|-- product_id: string (nullable = true)
|-- add_to_cart_order: string (nullable = true)
|-- reordered: string (nullable = true)
```

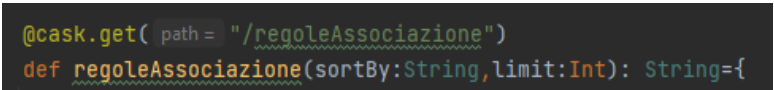
Figura 9: Schema di dati della tabella OrderProducts

order_id	product_id	add_to_cart_order	reordered
2	33120	1	1
2	28985	2	1
2	9327	3	0
2	45918	4	1
2	30035	5	0
2	17794	6	1
2	40141	7	1
2	1819	8	1
2	43668	9	0
3	33754	1	1
3	24838	2	1
3	17704	3	1
3	21903	4	1
3	17668	5	1
3	46667	6	1
3	17461	7	1
3	32665	8	1
4	46842	1	0
4	26434	2	1
4	39758	3	1

Figura 10: Prime 20 righe della tabella OrderProducts

2 Sviluppo e funzionalità back-end

Il lato back-end dell'applicativo è stato sviluppato, come da richiesta della traccia, sfruttando il framework Apache Spark. Per l'utilizzo delle sue API si è scelto di usare il linguaggio di programmazione Scala, preferito a Java e Python in quanto ha accresciuto le nostre conoscenze in merito ai linguaggi di programmazione ma al contempo ci ha permesso di sfruttare classi Java ove necessario. Si è scelto di adottare un'architettura di tipo client-server, in particolare il server espone delle REST API tramite cui il client può, mediante delle semplici richieste GET, ottenere dei dati in formato JSON. Il server è in ascolto continuamente sulla porta 6969, e può in ogni momento accogliere e servire delle richieste grazie al microframework open-source **Cask**, di semplice utilizzo grazie alle comode annotazioni che mappano i servizi a funzioni Scala.



```
@cask.get( path = "/regoleAssociazione")
def regoleAssociazione(sortBy:String,limit:Int): String={
```

Figura 11: Esempio di annotazioni Cask. Nel caso in cui il metodo associato al servizio richieda dei parametri, allora cask li andrà a valorizzare direttamente da omonimi parametri nell'URL

Ogni servizio REST, come menzionato in precedenza, produce come body della risposta HTTP dei dati in formato JSON. Per la trasformazione dei risultati delle operazioni sui DataFrame Spark in JSON abbiamo creato dei metodi ad hoc che costruiscono la stringa JSON. Ogni metodo risiede nell'oggetto Scala denominato *RestServices*

2.1 Descrizione Query

In totale nel file *RestServices.scala* sono state realizzate 9 queries, più un metodo ausiliario per la creazione della risposta JSON, di seguito ne si dà un elenco ed una spiegazione:

- **Top Prodotti:** essa è una query che, preso come parametro un numero intero $\langle x \rangle$, restituisce x coppie $\langle nomeProd, count \rangle$ che rappresentano rispettivamente nome e numero di vendite degli x prodotti più venduti.

- **Ore Giorno Acquisti:** essa è una query che non riceve alcun parametro ma restituisce un insieme di coppie $\langle ora, count \rangle$ che rappresentano, rispettivamente, una specifica ora del giorno (dunque un intero da 0 a 24) ed il numero di acquisti effettuati in quell'ora.
- **Giorni da Acquisto Precedente:** anch'essa non riceve parametri e restituisce un insieme di coppie $\langle numGiorni, count \rangle$ che rappresentano, rispettivamente, un numero di giorni dal precedente acquisto di un utente ed il numero di volte in cui un utente ha effettuato un acquisto esattamente $numGiorni$ giorni dopo l'ultimo.
- **Giorni della Settimana:** query non ricevente alcun parametro che restituisce all'utente un insieme di coppie $\langle dow, count \rangle$ dove dow è un intero compreso tra 0 e 6 rappresentante un giorno della settimana (da Domenica a Sabato) e $count$ è il numero di acquisti presenti nel dataset effettuati in quel giorno della settimana.
- **Numero Oggetti In Ordine:** questa query, non ricevendo parametri, restituisce un insieme di coppie $\langle numOggetti, count \rangle$ in cui il primo elemento è un numero di oggetti nel carrello, ed il secondo è un conteggio di quanti ordini nel dataset hanno esattamente $numOggetti$ prodotti nel carrello.
- **Dipartimenti Prodotti Venduti:** questa query, ultima a non ricevere parametri, restituisce al chiamante una lista di coppie $\langle reparto, count \rangle$ in cui $reparto$ è uno dei reparti interni di InstaCart e $count$ è il numero di prodotti venduti che sono afferenti al reparto in questione.
- **Regole Associazione:** questa query, ricevente in input una coppia di parametri $\langle sortBy, limit \rangle$ genera l'insieme delle prime $limit$ regole di associazione generate dall'algoritmo FPGrowth addestrato sul training set ordinate secondo la metrica indicata in $sortBy$. Questa query, dietro le quinte, effettua il training dell'algoritmo di Machine Learning FPGrowth (Frequent Pattern Growth). Esso è un algoritmo pensato per effettuare il cosiddetto **Association Rule Mining**, ovvero il processo di costruzione di delle relazioni di implicazione tra prodotti all'interno di una transazione. Molto semplicemente, una regola di associazione, posto che un utente abbia un prodotto A all'interno

del suo carrello, aiuta ad identificare un possibile prodotto B da comprare insieme ad esso. Nella definizione di regole di associazione si usano 3 principali metriche: **Support**, **Confidence**, **Lift**.

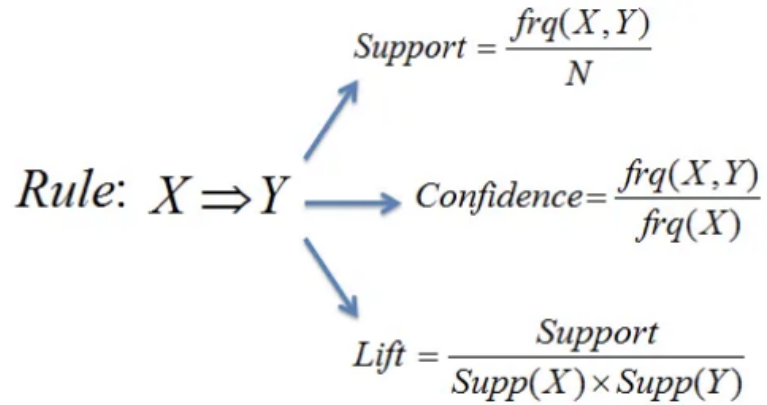


Figura 12: Le metriche delle Regole di Associazione

Il **Support** è definito come la divisione tra il numero di volte in cui gli oggetti X e Y sono comprati insieme ed il numero totale di transazioni, di fatti dicendo quanto è frequente il set di oggetti $\{X, Y\}$ all'interno dell'insieme delle transazioni.

La **Confidence** è definita come il numero di volte in cui i due oggetti X e Y sono comprati insieme rapportata al numero di volte in cui la testa dell'implicazione (dunque sono X) è acquistata.

Il **Lift** è definito come il Supporto della coppia $\langle X, Y \rangle$ diviso il prodotto dei supporti dei singoli oggetti. Se $Lift > 1$ ciò vorrà dire che i due oggetti sono dipendenti l'uno dall'altro, se $Lift = 1$ allora vuol dire che i due oggetti sono pressochè indipendenti l'uno dall'altro mentre se $Lift < 1$ vorrà dire che l'aver comprato X ha un impatto negativo sulla probabilità di comprare anche Y .

Nella query, dunque, una regola di associazione è una tupla del tipo:

$$\langle antecedent, consequent, confidence, lift, support \rangle$$

dove, rispettivamente *antecedent* è la lista di prodotti rappresentante la testa della regola di associazione (ciò che ho comprato), *consequent* è la lista di

prodotti rappresentante la coda della regola di associazione (ciò che è probabile che io compri posto che ho comprato *antecedent*). Nell'implementazione di MLLib (libreria di Machine Learning di Spark) dell'algoritmo FPGrowth si possono settare alcuni parametri il minimo support e la minima confidence, che sono stati settati in modo da non rendere l'elaborazione troppo lunga (a support più bassi corrispondono tempi di esecuzione molto alti).

- **Prodotti Comprati Insieme:** questa query riceve un parametro $\langle numProdotti \rangle$ e restituisce un insieme di coppie $\langle items, freq \rangle$ dove il primo elemento è un insieme di prodotti di taglia almeno $numProdotti$ ed il secondo è il numero di volte in cui quell'insieme di prodotti appare insieme in una transazione. Anche questa query fa uso del modello di cui si è discusso nel precedente punto della lista, in quanto al primo passo della sua esecuzione l'algoritmo FPGrowth identifica gli item frequenti.
- **Predizione:** questa è una query che, ricevendo una sequenza di stringhe rappresentante un possibile insieme di oggetti all'interno di un carrello, restituisce *suggestions* ovvero una lista di prodotti "suggeriti" da comprare insieme a quelli della sequenza. Ciò che avviene è che dalle regole generate in precedenza si filtrano quelle che non contengono gli elementi dello scontrino passato come input, e successivamente si prendono le prime 5 regole rimanenti ordinate per confidence, e si uniscono le loro code in un set, che sarà esattamente il *suggestions* restituito. L'idea alla base del funzionamento di questa query è che se trovo, tra le regole, una la cui testa contiene tutti gli elementi dello scontrino in input, allora posso supporre che la coda di tale regola sarà un buon suggerimento da dare all'utente.

3 Sviluppo del front-end ed estrazione della conoscenza

La parte grafica di visualizzazione dei dati è stata sviluppata utilizzando il framework open-source Flutter il quale sfrutta il linguaggio Dart. Il framework consente, tramite l'utilizzo di strutture chiamate "Widget", di visualizzare e navigare al meglio tra le query messe a disposizione dal back-end tramite servizi Rest.

3.1 Prodotti più acquistati

Questa pagina mostra due grafici: sulla sinistra un grafico piramidale che consente a colpo d'occhio di visualizzare quali sono i prodotti più acquistati dagli utenti di InstaCart e in che numero. Sulla destra un grafico a torta in cui gli stessi dati sono confrontati col numero dei prodotti rimanenti; questo per sottolineare l'enorme quantità di dati presenti all'interno del dataset.

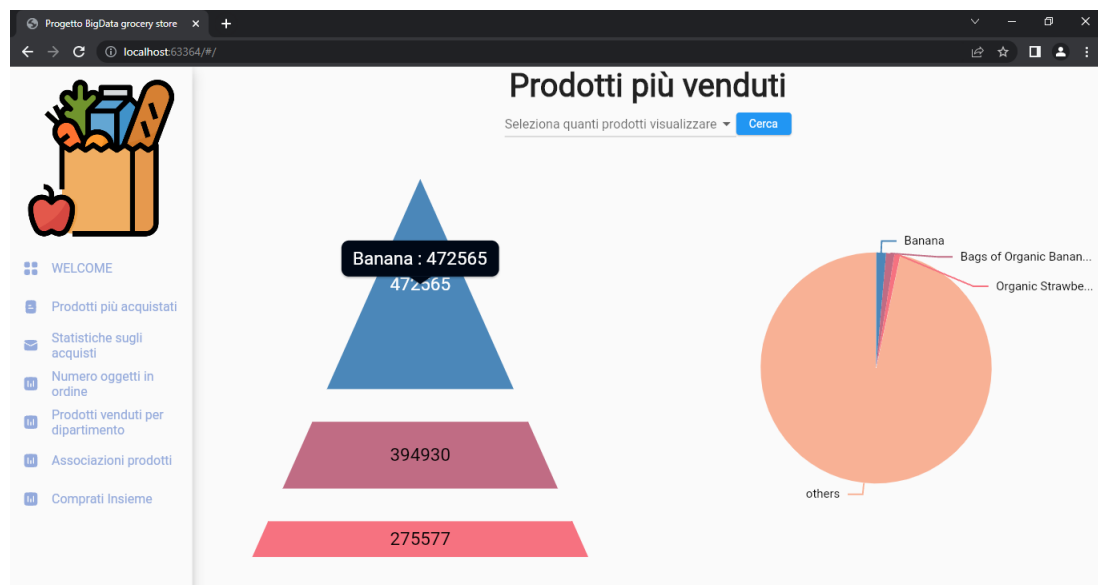


Figura 13: Schermata Prodotti più acquistati

Da questi grafici si evince che il prodotto più acquistato in assoluto è "Banana" con quasi mezzo milione di acquisti, seguito da "Bags of Organic Banana" e "Organic Strawberries". Per una ricerca più approfondita è possibile, tramite l'apposito menù a tendina fornito di pulsante di conferma, scegliere il numero di prodotti da visualizzare.

3.2 Statistiche temporali sugli acquisti

In questa sezione vengono mostrati i grafici relativi agli acquisti distribuiti nel tempo da parte degli utenti. In particolare è possibile visualizzare tramite un grafico cartesiano la quantità di prodotti acquistati nelle varie ore del giorno. Da qui si può facilmente dedurre che gli utenti tendono ad effettuare ordini in una fascia oraria compresa tra le 10 e le 16. Nel grafico a colonne verdi, similmente, si evince che gli utenti preferiscono ordinare i prodotti di Domenica e Lunedì mentre Giovedì è il giorno con meno ordini della settimana. Il grafico a colonne rettangolari blu mostra quanto spesso i clienti effettuano un ordine. La maggior parte dei clienti concludono un ordine una volta a settimana, questo si evince in quanto la maggior parte dei record sono concentrati tra 0 e 7 giorni. Inoltre, sempre in questo grafico, notiamo un picco al giorno 30; questo è dovuto in quanto la colonna "days_since_prior" è limitata a 30 dunque tutti gli ordini effettuati dopo 30 o più giorni sono racchiusi qui.

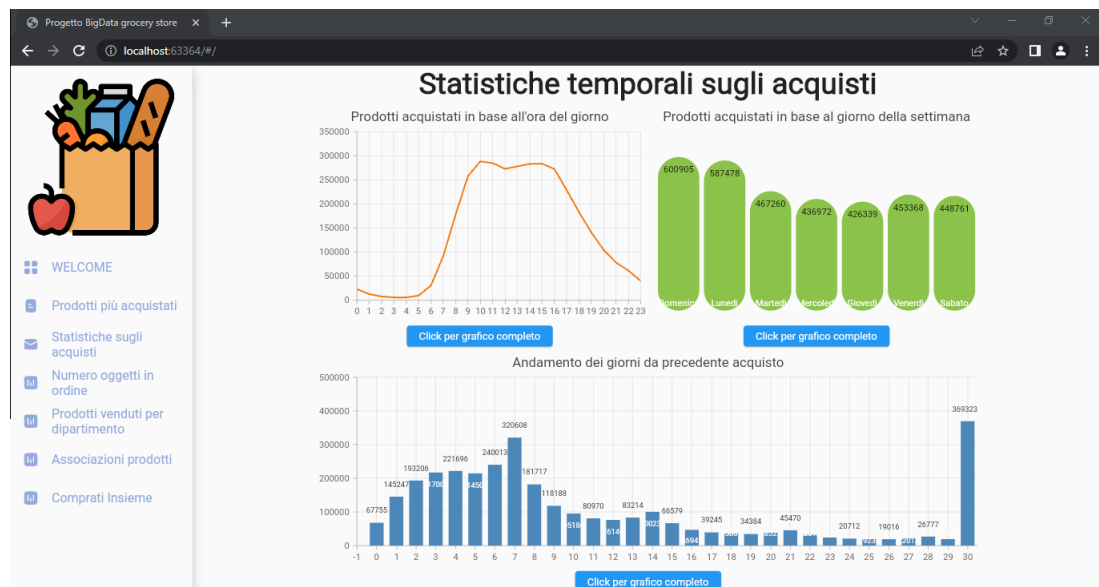


Figura 14: Schermata con le statistiche temporali

3.3 Numero oggetti in ordine

In questa schermata si vuole mostrare quanti prodotti vengono acquistati solitamente in un ordine.

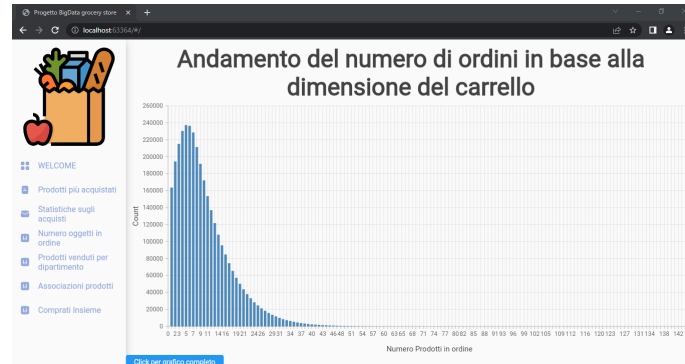


Figura 15: Andamento prodotti

Il grafico ci mostra che in media un utente fa un acquisto di 5 prodotti alla volta e, in generale, la maggior parte dei clienti preferisce un acquisto di un numero tra 1 e 15 prodotti per un singolo ordine. Ciò ci fa pensare che gli utenti di InstaCart preferiscano usarlo per effettuare spese medio/piccole. Inoltre, abbastanza sorprendentemente, troviamo una transazione di ben 145 prodotti diversi.

3.4 Prodotti venduti per reparto

Questa pagina si concentra sui reparti più apprezzati dagli utenti. Analizzando il grafico a torta si può subito notare come i reparti "produce" (frutta e verdura) e "dairy eggs" occupino quasi il 50% di tutti i reparti, seguiti da "snacks" e "frozen".

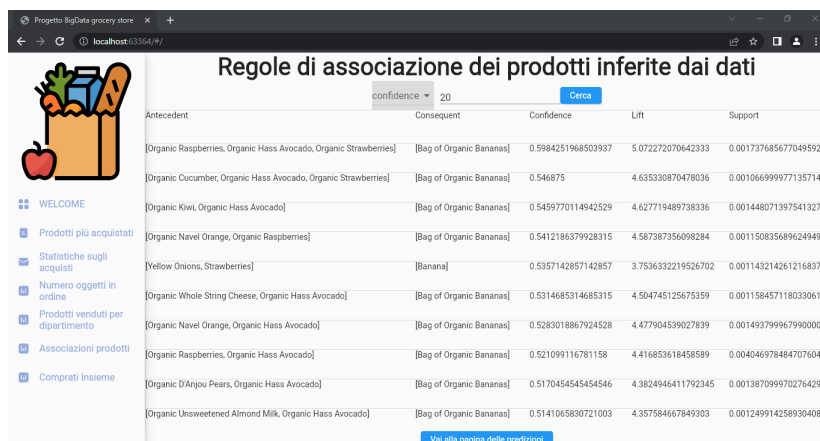


Figura 16: Reparti più richiesti

Questi dati sono abbastanza sorprendenti, in quanto InstaCart è un servizio principalmente usato negli Stati Uniti, uno dei paesi col tasso di obesità più alto del mondo, e dal grafico di evince che il reparto "snacks" abbia venduto meno della metà dei prodotti del ben più salutare reparto "produce" (stante per frutta e verdura),

3.5 Regole di associazione dei prodotti

In questa pagina vengono mostrati i risultati dell'algoritmo FPGrowth. Possiamo ordinare le associazione in base ai parametri di "confidence", "lift" e "support" tramite il menu a scomparsa in alto. Per ogni riga, sotto la colonna "antecedent" è mostrata la lista di prodotti nel carrello e sotto la colonna "consequent" il prodotto che un utente potrebbe acquistare con un percentuale dettata dallo storico degli acquisti ed espressa nella costante "confidence".



Antecedent	Consequent	Confidence	Lift	Support
[Organic Raspberries, Organic Hass Avocado, Organic Strawberries]	[Bag of Organic Bananas]	0.5984251968503937	5.072272070642333	0.0017376856770495927
[Organic Cucumber, Organic Hass Avocado, Organic Strawberries]	[Bag of Organic Bananas]	0.546875	4.635330870478036	0.0010669999771357147
[Organic Kiwi, Organic Hass Avocado]	[Bag of Organic Bananas]	0.5459770114942529	4.627719489738336	0.001448071397541327
[Organic Navel Orange, Organic Raspberries]	[Bag of Organic Bananas]	0.5412186379928315	4.587387356098284	0.0011508356896249496
[Yellow Onions, Strawberries]	[Banana]	0.5357142857142857	3.7536332219526702	0.0011432142612168373
[Organic Whole String Cheese, Organic Hass Avocado]	[Bag of Organic Bananas]	0.5314685314685315	4.504745125675359	0.0011584571180330617
[Organic Navel Orange, Organic Hass Avocado]	[Bag of Organic Bananas]	0.5283018867924528	4.477904539027839	0.0014937999679900007
[Organic Raspberries, Organic Hass Avocado]	[Bag of Organic Bananas]	0.521099116781158	4.416853618458589	0.004046978484707604
[Organic D'Anjou Pears, Organic Hass Avocado]	[Bag of Organic Bananas]	0.5170454545454546	4.3824946411792345	0.0013870999702764292
[Organic Unsweetened Almond Milk, Organic Hass Avocado]	[Bag of Organic Bananas]	0.5141065830721003	4.357584667849303	0.0012499142589304088

Figura 17: Schermata con parametro "confidence" e limit 20

Dai risultati ottenuti si evince che se un cliente ha nel carrello "organic raspberries", "organic avocados" e "organic strawberries" allora è probabile che acquisti anche delle "organic bananas" con una confidence quasi al 70%.

3.6 Prodotti comprati insieme

In questa sezione dell'applicazione si vogliono mostrare i gruppi di prodotti maggiormente comprati insieme. Al solito dal menù a tendina è possibile scegliere la cardinalità dei gruppi.

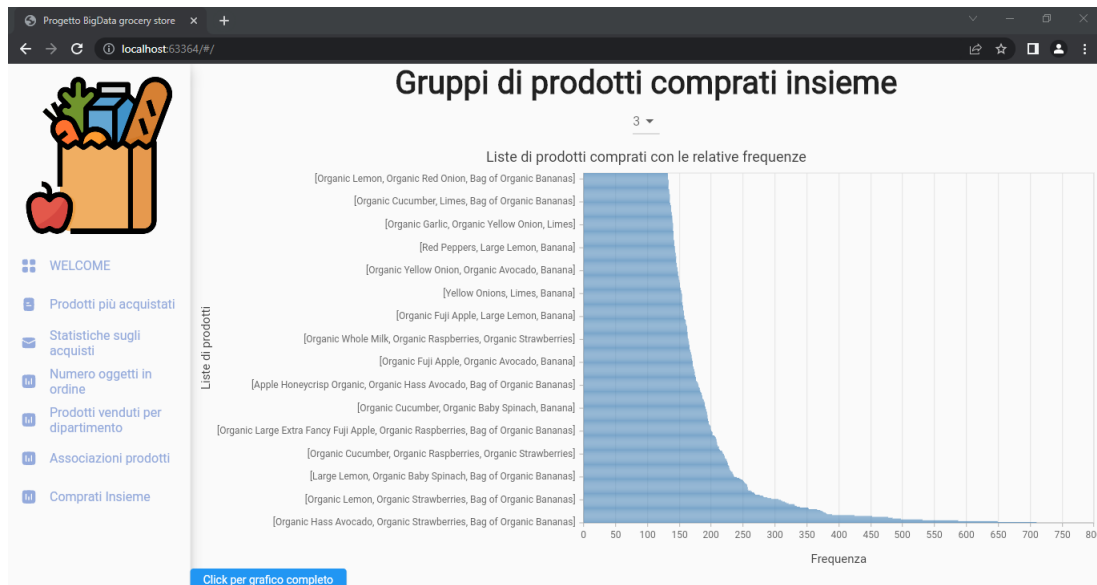


Figura 18: Schermata con parametro 3

Eseguendo la query con lunghezza 3 si può notare come la tripla "organic hass avocado", "organic strawberries", "bag of organic banana" sia la più frequente. Questi risultati risultano essere abbastanza "telefonati", in quanto sia dalla classifica "individuale" dei prodotti, che dalla classifica dei reparti, frutta e verdura l'hanno fatta da padroni. Risulta, dunque, poco sorprendente vedere come gli insiemi di prodotti comprati insieme più frequentemente sono formati da prodotti afferenti a queste categorie.