POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

ANALISI DEL COMPORTAMENTO DEI CLIENTI DI UN E-COMMERCE PER PREVEDERE LE INTENZIONI D'ACQUISTO

Relatore

Luigi DE RUSSIS

Candidato

Simone SINAGRA

Tutor Aziendale

Zero11

Michele SONNESSA

Ottobre 2021

Sommario

La User Experience (UX) in un sito di e-commerce è fondamentale per favorire la conversione dei visitatori in compratori, per massimizzare le vendite e per garantire la soddisfazione i clienti. L'intelligenza artificiale permette di analizzare i dati di navigazione degli utilizzatori dei siti e-commerce identificando comportamenti di acquisto omogenei. L'ipotesi di questo progetto di ricerca sperimentale è che utilizzando tecniche di intelligenza artificiale sia possibile realizzare una analisi prescrittiva dei comportamenti di acquisto che possa essere utilizzata in futuro per realizzare customer journey e customer experience (CX) personalizzate ed ottimizzate attraverso la modifica dinamica degli elementi di UI e UX con cui il visitatore interagisce.

Osservando ogni fase del Customer Journey si ha una rappresentazione dei comportamenti e delle necessità dei propri clienti. Un'analisi accurata della CJM permette di trovare potenziali punti di frizione che possono essere risolti per migliorare l'interazione dell'azienda con il pubblico e quindi di incrementare la soddisfazione nell'esperienza d'acquisto dei clienti. La naturale conseguenza di un miglior processo di acquisto è quella di avere un maggior numero di clienti ed un miglior tasso di conversione da visitatori a compratori. La soddisfazione, la fiducia e la fidelizzazione dei clienti sono tre fattori di spiccata importanza per poter creare un business di successo.

Ringraziamenti

Questo progetto di tesi segna la fine di cinque fantastici anni al Politecnico di Torino, in cui ho avuto modo di maturare e accrescere la mia conoscenza.
Desidero ringraziare il Prof. Paolo Cortese che ho avuto la fortuna di poter conoscere al mio primo esame universitario e il Prof. Giovanni Malnati che mi ha insegnato molto durante gli anni di magistrale. Un ringraziamento va al Prof. Edgar Ernesto Sanchez Sanchez che per un semestre intero mi ha fatto appassionare in un ambito dell'informatica che pensavo non fosse di mio interesse e a tanti altri professori che mi hanno guidato durante l'intero percorso universitario.
Vorrei anche ringraziare tutti i miei compagni di università con cui ho condiviso le mie giornate al Politecnico, e ne approfitto per augurare a tutti un grosso in bocca al lupo per il loro futuro.
Un ringraziamento speciale va a tutte le persone che, in prima linea, mi hanno aiutato durante l'intero processo di stesura della tesi: al relatore accademico, il Prof. Luigi De Russis, per la sua disponibilità, la sua gentilezza e l'attenzione che ha dimostrato durante questi mesi; a tutte le persone che mi hanno accolto presso l'azienda Zero11, in particolare ad Andrea e Michele che con i loro consigli si sono rivelati un valore aggiunto per lo sviluppo di questo progetto.
Grazie alla mia famiglia, mia mamma, mio papà e mia sorella che mi hanno sempre supportato per tutto il percorso della mia crescita.
Grazie a Sara che è sempre stata al mio fianco.
Senza di loro non sarei la persona che sono e sono sicuro che su di loro potrò fare sempre affidamento per poter raggiungere molti altri traguardi importanti della mia vita.

Simone

Indice

Elenco delle tabelle

Elenco delle figure

Acronimi

API
Application Programming Interface

CAGR
Compound Annual Growth Rate

CJ
Customer Journey

CJM
Customer Journey Map

CRM
Customer Relationship Management

CTA
Call-to-Action

CX
Customer Experience

DBMS
Data Base Management System

ETL
Extract, Transform, Load

GDS
Graph Data Science

KDD
Knowledge Discovery from Data

URI
Universal Resource Identifier

URL
Uniform Resource Locator

UX
User Experience

Capitolo 1

Introduzione

1.1 Contesto

La digitalizzazione ha cambiato in modo radicale molti aspetti della nostra società e molte aziende hanno sfruttato la rapida crescita di Internet per creare nuove fonti di guadagno, costruendo modelli di business innovativi. La nascita del commercio elettronico ha portato a cambiamenti nella vendita di prodotti e servizi e ha generato nuove opportunità e sfide per le aziende, che hanno dovuto adattare le loro strategie per far fronte a questa innovazione e per essere competitive su un mercato sempre più in crescita.

Come emerge dal report pubblicato da Casaleggio Associati nel 2021 [1], il settore dell'e-commerce in Italia ha registrato una crescita impressionante negli ultimi anni, raggiungendo un record di fatturato nel 2019, pari a 48.5 miliardi di euro, con un Compound Annual Growth Rate del 25.5% nel periodo 2004-2019. Il business online si è poi stabilizzato nel 2020, con una leggera decrescita dell'1%, dovuta alla diffusione della pandemia globale SARS-CoV-2, che ha fortemente colpito alcuni settori, come il turismo, tra i principali responsabili per le vendite online, favorendo però la crescita di altri, come l'alimentare. Nonostante ciò, la pandemia ha dato una spinta considerevole al processo di digitalizzazione dei canali di vendita di molte imprese. Infatti, nel solo 2020, sono entrate nel mercato 10.467 nuove imprese che si sono registrate al registro imprese con codice ATECO 47.91.1 relativo al commercio online, che si traduce in un aumento del 50% in un solo anno.

Lo sviluppo di Internet ha influito molto anche sui consumatori. Dai dati del report [1] si osserva che è sempre più in aumento il numero di utenti che accedono al web, con una stima di 4.6 miliardi nel 2020 (+7% rispetto all'anno precedente), che rappresenta il 59% della popolazione mondiale. Inoltre, la pandemia ha cambiato

le aspettative dei clienti nei confronti dei negozi, che devono sempre più cercare di soddisfare le loro esigenze offrendo delle esperienze di pari valore a quelle nei negozi fisici.

L'esigenza delle aziende che si interfacciano con il mondo digitale è quella di creare un business in grado di garantire ai clienti esperienze d'acquisto comparabili o migliori rispetto alle esperienze nei negozi fisici. Una delle strategie più diffuse e ricercate nel mondo digitale, specialmente nell'ambito degli e-commerce, è quella di creare contenuti dinamici, ovvero contenuti che si adattano in base al cliente che sta utilizzando il sito web. Le modifiche alle pagine visualizzate dai clienti possono variare in base a diversi fattori del consumatore, tra cui le sue caratteristiche demografiche, i suoi comportamenti durante una sessione oppure in base a sue interazioni passate con l'azienda.

L'utilizzo di contenuti dinamici è uno strumento che può garantire un incremento di conversioni rendendo la Customer Experience più rilevante e coinvolgendo maggiormente i clienti, con un conseguente aumento delle entrate e aumento della fedeltà dei clienti.
Il concetto di contenuti dinamici è sempre esistito, anche prima dell'avvento di Internet, e in un certo senso viene applicato in molteplici situazioni della vita reale, nei negozi fisici. Prendiamo come esempio una panetteria. Il panettiere ogni giorno si relaziona con un numero finito di clienti e cerca di creare un legame con le persone che visitano il suo negozio, cercando di ricordare le loro preferenze ed abitudini. Con il passare del tempo si presenteranno situazioni in cui saprà già di cosa necessitano i suoi clienti, velocizzando delle operazioni che potrebbero essere superflue, saprà cosa consigliare a clienti che reputa simili ad altri clienti passati, così come saprà che per esempio un cliente entra in negozio con il suo piccolo figlio a cui piace tanto un prodotto e quindi nel sacchetto della spesa gli ripone questo un assaggio di questo prodotto, e così via... Sono tutte azioni che risultano personalizzate per un singolo cliente o per un segmento di clienti e sono azioni che permettono di migliorare il rapporto con i clienti. La naturale conseguenza di un rapporto di questo tipo è quella di aumentare il numero i clienti (ed aumentare la probabilità che essi ritornino). Inoltre, in un negozio fisico si potrebbero per esempio anche creare delle offerte scrivendo su un cartello esposto all'esterno ma in questo caso sarebbe difficile poter differenziare e personalizzare l'offerta in base al cliente perché il cartello lo leggerebbe chiunque.

Tuttavia ci sono anche alcuni aspetti negativi nell'utilizzo di contenuti dinamici: oltre all'aumento di complessità si deve far fronte anche a un potenziale caricamento più lento delle pagine, per questo motivo è necessario scegliere in modo accurato dove si può trarre maggiore vantaggio con l'utilizzo di contenuti personalizzati.

Per esempio la home page potrebbe essere una delle pagine che più beneficiano dei vantaggi dei contenuti dinamici perché di solito è la prima pagina che viene mostrata a un potenziale cliente (e potrebbe essere inutile mostrare certi contenuti), oppure la pagina di un prodotto che potrebbe richiedere maggiori dettagli per un cliente che non ha mai acquistato quel prodotto rispetto a un cliente che ha già acquistato lo stesso prodotto.

1.2 Obiettivo

Il contesto descritto nella sezione 1.1 mette in risalto quanto sia importante per un'azienda curare l'interazione e il rapporto con i clienti per poter crescere in un mercato sempre più competitivo. L'utilizzo di canali digitali permette di raccogliere, attraverso l'uso di strumenti di monitoraggio, una grande quantità di dati durante la navigazione dei clienti sul sito web. La CJM permette di analizzare lo stato d'animo di un cliente durante il suo processo di acquisto ma è difficile riuscire ad analizzare nel complesso tutti i touchpoint e la soddisfazione del cliente durante le interazioni (online e offline).

In un negozio fisico un venditore si relaziona direttamente con i clienti, che possono offrire degli indizi fondamentali. Un buon venditore può essere in grado di percepire le intenzioni e gli interessi di un cliente in base ai suoi modi di fare, di muoversi o camminare, in base al suo abbigliamento o per esempio in base alle domande che esso gli pone. Con la digitalizzazione l'audience può raggiungere un numero molto elevato ed entra in gioco il concetto di Big Data e le fonti da cui si possono ricavare indizi sui clienti possono essere di diverso tipo, per esempio: strumenti di monitoraggio e analisi, feedback tramite sondaggi e molto altro...

Per questo progetto di tesi il focus principale è sulla fase centrale di acquisto attraverso il canale digitale, ovvero la fase in cui il cliente si relaziona direttamente con l'azienda utilizzando il sito e-commerce. L'obiettivo è quello creare un modello di classificazione dei clienti attraverso degli algoritmi di machine learning sfruttando le tracce che essi producono durante la navigazione nel sito web. Il modello generato sarà poi utilizzabile per poter modificare i contenuti in modo dinamico. Per esempio potrebbe essere un'ottima strategia quella di mostrare delle Call-to-Action particolari per clienti non intenzionati a comprare, oppure guidare velocemente all'acquisto un cliente che è intenzionato a comprare.

Tra i risultati attesi, si auspica la generazione di un buon modello predittivo di classificazione dei clienti come compratori o meno. Data l'enorme quantità di dati

a disposizione un focus particolare sarà posto anche sull'efficienza e correttezza dell'algoritmo per memorizzare i dati in un database non relazionale, al fine di garantire un'efficienza sufficiente per poter aggiornare periodicamente la base dati.

1.3 Struttura del documento

Il documento inizia con un'introduzione dell'ambito di ricerca, in cui si evidenziano alcuni dati che motivano l'interesse allo sviluppo di questo progetto di tesi.

Nel capitolo 2 "Concetti generali" si introducono le nozioni principali degli argomenti trattati nella tesi. È presente una prima sezione in cui si introduce il concetto di e-commerce, si descrivono le metodologie per poter studiare il customer journey e si sottolinea l'importanza dell'offrire contenuti dinamici ai clienti per poter migliorare la conversione in compratori e la soddisfazione generale nell'uso dell'e-commerce. Sono presenti inoltre due sezioni su concetti più pratici, quali i database e la data science. Nella prima sezione si confrontano le varie tipologie di basi di dati, mentre nella seconda sezione si discute dell'avvento dei big data e delle tecniche per poterli analizzare.

Il capitolo 3 "Metodologia" è utilizzato per poter descrivere l'approccio utilizzato per sviluppare il modello di classificazione, specificando il punto di partenza e ciò che si vuole ottenere.

Il capitolo 4 "Salvataggio delle visite in una struttura a grafo" e il capitolo 5 "Implementazione del modello predittivo" sono i capitoli principali in cui si descrivono tutte le azioni pratiche usate per ottenere il risultato atteso. Nel primo capitolo si procede con il trasferimento dei dati offerti da un software di tracking in una base di dati a grafo, mediante un complesso processo di preparazione dei dati. Nel secondo capitolo invece si utilizzano i dati che sono stati salvati precedentemente nella base di dati. Per poter generare il modello predittivo si è eseguita una fase di pre-processing dei dati e sono stati selezionati accuratamente i migliori parametri per poter eseguire l'algoritmo di machine learning per la classificazione delle visite.

Infine, il capitolo 6 conclude il documento riassumendo gli obiettivi raggiunti, offrendo alcuni spunti per poter utilizzare il modello ricavato e descrivendo eventuali lavori futuri.

Capitolo 2

Concetti generali

Questo capitolo fornisce un'introduzione sui principali temi trattati in questo progetto di tesi. La prima sezione descrive i concetti principali degli e-commerce: i diversi modelli di business, l'importanza dello studio del customer journey e dell'uso dei contenuti dinamici. Successivamente è presente un confronto tra i diversi tipi di database, con un particolare attenzione alle basi dati a grafo, utilizzate nei capitoli successivi. Infine, dopo una breve introduzione ai big data, vengono descritti i diversi algoritmi di machine learning.

2.1 E-commerce

L'insieme delle attività di acquisto e vendita di prodotti e servizi tramite Internet prende il nome di e-commerce. Esistono diversi modelli di commercio elettronico che variano in base ai soggetti coinvolti nell'attività di vendita/acquisizione. Ciascuno di questi modelli di business sfrutta a pieno i vantaggi di Internet, tra cui la possibilità di rendere accessibili i prodotti e i servizi al consumatore in qualsiasi momento e da qualsiasi parte del mondo, garantendo una clientela più ampia rispetto ad un negozio fisico.

2.1.1 Modelli di business

Tra i principali modelli applicati nell'ambito degli e-commerce [2] ci sono:

- **Business to Business (B2B)**: il modello B2B si occupa della vendita di prodotti e servizi da un'azienda all'altra. Un caso in cui si osserva questo modello è quello in cui l'azienda acquirente è l'intermediaria tra l'azienda produttrice e il consumatore finale.

- **Business to Consumer (B2C)**: nel commercio B2C la vendita di un prodotto è destinata direttamente al cliente finale. Questo modello è la rappresentazione digitale della tradizionale vendita al dettaglio in un negozio fisico.
- **Customer to Customer (C2C)**: un sito di e-commerce C2C consente ai clienti di vendere prodotti direttamente ad altri clienti, solitamente in cambio di una commissione per il servizio.
- **Customer to Business (C2B)**: questo tipo di business è l'inverso del modello B2C e prevede che sia il consumatore ad offrire i suoi prodotti o servizi alle aziende. In questo caso l'azienda è il customer e sfrutta la visibilità di un'altra azienda per ampliare la propria clientela.

A seconda del tipo di business di una azienda cambiano le strategie di marketing e la tipologia dei clienti e quindi il loro modo di interagire con l'e-commerce.

2.1.2 Customer Journey

Oggi, la strategia principale che le aziende utilizzano per gestire le relazioni e le interazioni con i clienti o potenziali tali è il Customer Relationship Management (CRM). Nel settore del CRM, si definisce Customer Journey (CJ) il percorso che un cliente, o un potenziale cliente, compie attraversando diversi punti di contatto, comunemente chiamati touchpoint. In altre parole, il CJ è l'insieme di tutte le interazioni che un cliente ha con l'azienda durante il suo "viaggio" sul sito e-commerce. Si parla inoltre di Customer Journey omnicanale quando l'esigenza dei clienti di arrivare a un prodotto viene soddisfatta attraverso diverse piattaforme e modalità. In questo caso i touchpoint possono essere sia online che offline, ovvero le interazioni possono iniziare e terminare su canali diversi (fisici o digitali). Per questo esistono diverse strategie di vendita, per esempio la vendita online con il ritiro del prodotto in negozio (Click & Collect), la prova di un prodotto nel negozio e il successivo acquisto online (Try & Buy), oppure la ricerca di un prodotto online e l'acquisto in negozio (Research Online, Purchase Offline).

A differenza del commercio esclusivamente nei negozi fisici, l'utilizzo di un canale di vendita digitale permette di avere a disposizione un'enorme quantità di informazioni dettagliate delle azioni che i clienti eseguono quando interagiscono con il sito e-commerce di un'azienda. Allo stesso tempo, a causa dei molteplici touchpoint in canali diversi, diventa difficile comprendere il comportamento dei clienti e di conseguenza aumenta la complessità nel valutare il Customer Journey e quindi migliorare la Customer Experience.
Un articolo di McKinsey & Company [3] illustra uno dei uno dei possibili modelli per descrivere il processo d'acquisto dei clienti. Tale modello è rappresentato sotto

forma di imbuto, in cui sono presenti cinque fasi fondamentali (mostrati in Figura 2.1):

Consapevolezza In questa fase il cliente ha un'esigenza e cerca un modo per soddisfarla. Questa è la fase iniziale d'acquisto (l'estremità più larga dell'imbuto) nel quale i consumatori prendono in considerazione una serie di potenziali aziende diverse. La consapevolezza dell'esistenza di diversi marchi per soddisfare il loro bisogno può avvenire attraverso diversi canali, per esempio tramite i social media o una ricerca su un motore di ricerca. L'obiettivo delle aziende è quello di attrarre il più alto numero di potenziali clienti all'ingresso di questo imbuto.

Familiarità Questa fase ha come obiettivo principale quello di costruire una certa familiarità con il brand, analizzando a quale mercato di riferimento appartiene e quali prodotti propone.

Considerazione Dopo che un cliente è consapevole dell'esistenza di un marchio e ne dimostra un certo interesse inizia la terza fase, in cui i clienti raccolgono informazioni sui prodotti per valutare l'acquisto. In questa fase è importante rimuovere ogni dubbio ai clienti, per esempio descrivendo in modo accurato i prodotti, fornendo testimonianze di altri clienti... È fondamentale offrire una certa semplicità e fluidità nell'eseguire ogni azione del processo di acquisto sul sito e-commerce.

Acquisto L'acquisto coincide con la fase finale del processo di vendita. In questa fase l'azienda è riuscita a incanalare il cliente fino alla fine dell'imbuto. È importante mettere a disposizione del cliente una pagina per effettuare il pagamento ben progettata, per esempio offrendo più opzioni di pagamento.

Fedeltà La fase precedente è solo il primo dei due obiettivi principali delle aziende, in quanto è fondamentale creare un rapporto di continuità con il cliente, in modo da favorire il suo ritorno sul sito e quindi aumentare la possibilità di vendergli nuovi prodotti. Questa fase è cruciale per il processo di crescita dell'azienda.

Figura 2.1: Modello tradizionale del Customer Journey

In considerazione al fatto che il cliente può interagire con un'azienda più volte (ad esempio effettuando molteplici acquisti in momenti diversi), il modello ad imbuto, per quanto sia valido per rappresentare in modo lineare le varie fasi del processo di acquisto, viene superato da un modello ciclico dove gli step fondamentali rimangono gli stessi ma si evidenzia il fatto che le interazioni con l'azienda sono multiple e la fine di ogni ciclo influenza l'inizio del successivo.
Nel Customer Journey la fase di valutazione e scrematura dei brand avviene in modo analogo al modello ad imbuto, ma in seguito all'acquisto, il cliente fonda delle aspettative che saranno fondamentali per decisioni di acquisto future. Infatti, nella fase iniziale di acquisto, in cui un cliente seleziona un insieme di brand da cui partire per soddisfare una necessità, saranno fondamentali le esperienze passate e touchpoint con le aziende recenti. Lo stesso articolo di McKinsey & Company [3] chiama questo tipo di modello "processo decisionale" (rappresentato in figura 2.2). Una descrizione analoga viene fornita nella pubblicazione di Katherine N. Lemon & Peter C. Verhoef [4].

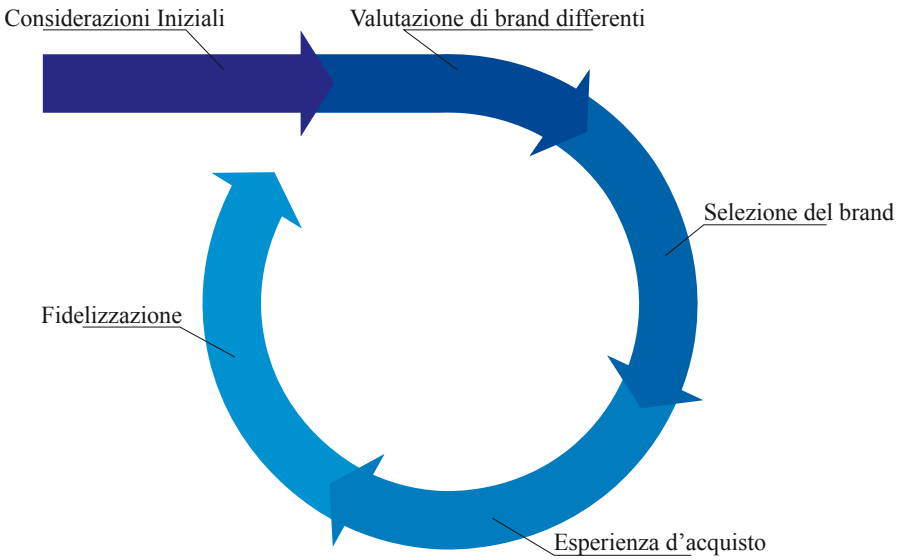

Figura 2.2: Modello circolare del percorso decisionale di un cliente

Nella pubblicazione di R. Mangiaracina et al [5] vengono identificati cinque passi fondamentali della UX durante l'interazione con un sito e-commerce per l'acquisto di uno o più prodotti:

1. **Ingresso sul sito**: questa fase si focalizza su come gli utenti raggiungono il sito e-commerce. Siccome l'esperienza comincia fuori dal sito e-commerce, è importante valutare le modalità utilizzate per poter attrarre un potenziale cliente a visitare il sito, per esempio utilizzando un motore di ricerca o attraverso la newsletter. In questa fase è importante curare il primo impatto del cliente con il sito, per esempio garantendo un rapido caricamento della pagina (di solito la homepage) oppure offrendo la possibilità di trovare ciò di cui ha bisogno senza dover scorrere nella pagina.
2. **Navigazione nel catalogo e scoperta di prodotti**: in questa fase è importante focalizzarsi su come gli utenti trovano i prodotti sul sito e-commerce. Ci sono diversi approcci di ricerca di un prodotto. Un metodo è quello in cui il cliente conosce già il prodotto che sta cercando e quindi utilizza direttamente il motore di ricerca oppure effettua una ricerca a partire dalle categorie. Un altro approccio è quando il cliente non conosce il prodotto preciso ma ha un'idea sulle caratteristiche che deve avere, in questo caso il cliente deve fare una ricerca tra diversi prodotti e bisogna evitare che si perda tra troppi prodotti che non sono di suo interesse.
3. **Presentazione del prodotto**: la presentazione del prodotto è fondamentale per influenzare il cliente nella sua decisione. Il cliente non può vedere fisicamente il prodotto perciò è necessario presentarlo con precisione con delle

descrizioni precise e fornendo delle immagini.
4. **Gestione del carrello**: in questa fase è prevista una gestione della lista della spesa generata durante la navigazione sul sito e-commerce. Si tratta di una fase critica in quanto presenta un tasso di abbandono molto elevato.
5. **Configurazione dell'ordine e processo di checkout**: questa ultima fase avviene nel momento in cui il cliente ha deciso di voler completare i suoi acquisti. Questo processo deve essere completato nel modo più semplice e più breve possibile per evitare di perdere il cliente proprio nel momento conclusivo. Tra i parametri per valutare la complessità di questa fase ci sono il numero di click, complessità dei moduli web, fasi del processo, acquisto con o senza registrazione...

2.1.3 Customer Journey Map

Per poter analizzare un Customer Journey è necessario rappresentare ciascuna fase di interazione tra il cliente e l'azienda. Uno strumento molto utilizzato sono le Customer Journey Map. Il formato di una CJM solitamente è caratterizzato da una parte superiore in cui viene descritto l'utente (o un segmento di utenti), uno scenario e le aspettative/obiettivi; una parte centrale contenente le fasi del "viaggio", con azioni, pensieri e emozioni dell'utente; una parte inferiore con opportunità e intuizioni, per evidenziare possibili ottimizzazioni del percorso del cliente.

Una Customer Journey Map è definita come una serie di obiettivi e azioni dell'utente disposte seguendo l'ordine temporale in cui avvengono e sono arricchite con i pensieri e le emozioni dell'utente al fine di creare una storytelling [6].

Nel dettaglio, la mappa del Customer Journey viene decomposta tipicamente in tre zone, come mostrato in Figura 2.3:

- Zona A: (1) assegnazione di una persona (chi) e (2) descrizione dello scenario da esaminare (cosa).
- Zona B: (3) fasi del journey, (4) azioni, (5) pensieri e (6) esperienza emotiva dell'utente
- Zona C: (7) intuizioni, punti deboli e quindi le opportunità scoperte, (8) internal ownership (chi sarà responsabile delle modifiche, in base alle opportunità identificate)

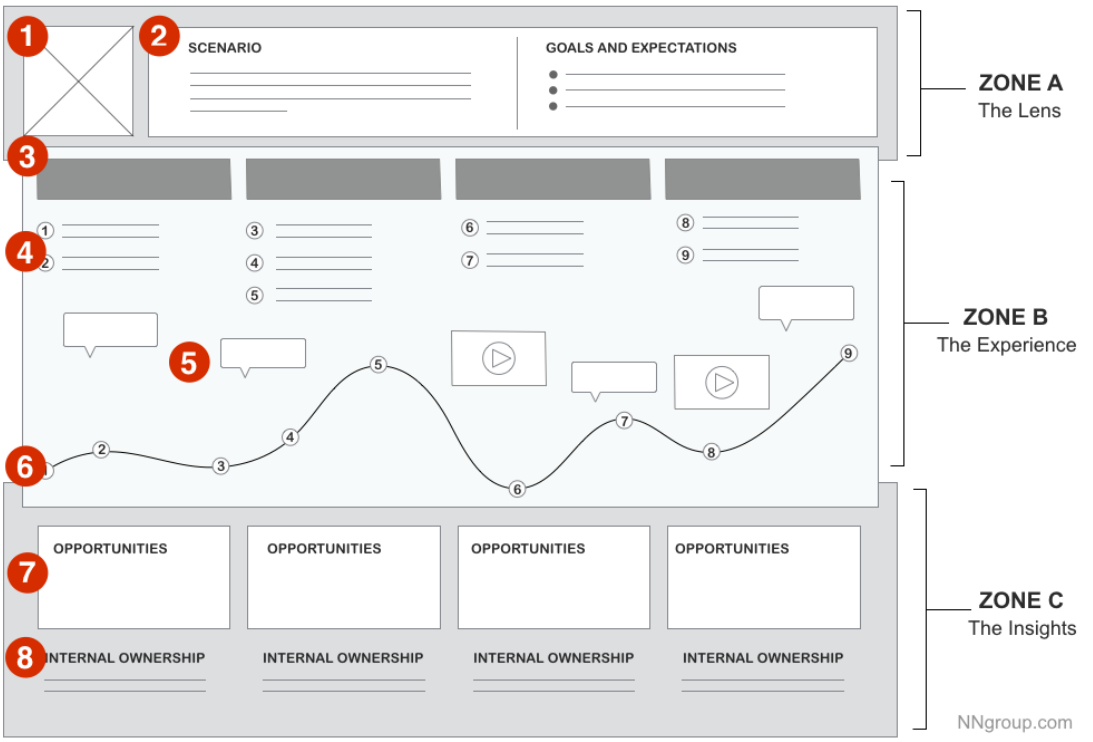

Figura 2.3: Modello generale di una Customer Journey Map [6]

L'articolo pubblicato su NN/g Nielsen Norman Group [7] offre sette metodi per poter analizzare una CJM in modo da trovare opportunità di miglioramento del processo di vendita, tra cui quello di cercare touchpoint superflui; oppure quello di ricercare punti nel percorso in cui le aspettative non sono rispettate (le aspettative potrebbero essere causate da esperienze passate); oppure ancora quello di valutare il tempo speso in ogni stadio fondamentale del processo di vendita. Oltre a ricercare punti negativi è anche importante valutare punti nel percorso in cui il cliente è soddisfatto e le aspettative vengono rispettate per cercare di replicare queste situazioni in altri momenti.

2.1.4 Contenuti web dinamici

Catherine Vanvonno [8] descrive quattro idee di contenuto dinamico per aumentare il numero di conversioni e-commerce:

- Inviare email personalizzate: a differenza dei messaggi generici, un messaggio specifico per un particolare cliente o un determinato segmento di clienti garantisce un tasso di apertura e lettura delle email maggiore.

- Migliorare la ricerca: la ricerca è un elemento importante durante la navigazione e può avere un grosso impatto sul tasso di soddisfazione, perciò richiede particolare attenzione. La ricerca non deve essere solo uno spazio in cui digitare parole, ma deve essere arricchita dalle funzioni di completamento e correzione automatica. I suggerimenti devono essere pertinenti al comportamento del cliente sul sito.
- Gestire correttamente i carrelli abbandonati: le statistiche di uno studio condotto da Barilliance [9] mostra che più dei 3/4 dei clienti abbandonano il sito senza completare l'acquisto. Il numero elevato di abbandono richiede particolare attenzione e una delle possibili contromisure che si adottano è quella di inviare dei messaggi per ricordare ai clienti i loro acquisti non portati a termine, includendo un link per poter proseguire all'acquisto.
- Uso di Call-to-Action persuasive: una CTA è una strategia per richiedere all'utente di eseguire un'azione. Solitamente si tratta di azioni importanti nel processo di vendita e perciò necessitano di una particolare attenzione per poter aumentare il tasso di conversione. Un esempio è quello di utilizzare pop-up per mostrare un'offerta personalizzata in base alla navigazione del cliente.

L'articolo di Komal Helyer [10] propone otto esempi concreti di utilizzo di contenuti dinamici, tra cui:

- Timer che mostra un countdown di un evento, per esempio per segnare l'inizio o la fine di una vendita. I timer possono anche essere creati in modo specifico per un determinato cliente e solitamente provocano un senso di urgenza che può velocizzare la conversione.
- Raccomandazione dei prodotti, consentono di mostrare contenuti diversi in base al cliente che sta utilizzando il sito. Le raccomandazioni personalizzate per esempio possono avvenire anche per proporre un completamento del look in base ai prodotti inseriti nel carrello, Inoltre questo metodo permette di conoscere meglio il cliente.
- Coupon o Voucher personalizzati garantiscono un rapporto migliore con il cliente, specialmente se si tratta di offerte personalizzate.
- Banner dinamici per personalizzare le pagine visitate dai clienti in base.
- Prezzi e disponibilità in tempo reale provocano anch'essi un senso di urgenza e sono particolarmente utilizzati dalle aziende di viaggi in cui i prezzi e le disponibilità variano regolarmente.

2.2 Persistenza dei dati

Nell'ambito dell'informatica, molte applicazioni richiedono di poter memorizzare i dati in apposite strutture per poterci accedere in modo efficiente nel tempo. Tali strutture prendono il nome di base di dati (DB) e rappresentano una collezione di dati memorizzati in modo persistente su un supporto fisico gestito da un Data Base Management System (DBMS), un sistema software che garantisce l'affidabilità e la privatezza di un DB [11].

Una base dati può avere dimensioni elevate, perciò il DBMS deve garantire il corretto funzionamento avendo come unico limite la dimensione fisica di memorizzazione. I DBMS devono inoltre offrire dei metodi di accesso concorrente alla base dati in quanto esse sono il più delle volte condivise da più applicazioni.

Un DBMS permette di eseguire diverse operazioni, relative alla struttura della base di dati e ai record. Tali operazioni sono descritte da istruzioni scritte in uno specifico linguaggio per le basi di dati, che si dividono in due categorie:

- **Data Definition Language (DDL)**: linguaggio utilizzato per la definizione dei dati e per autorizzare l'accesso alla base di dati;
- **Data Manipulation Language (DML)**: linguaggio che definisce le istruzioni inserire, rimuovere o modificare i record memorizzati nella base di dati;
- **Data Query Language (DQL)**: linguaggio che permette di recuperare i dati memorizzati nella base di dati.

2.2.1 Basi di dati relazionali

Le basi di dati relazionali si basano sul modello relazionale, descritto da E.F. Codd in una pubblicazione del 1970 [12]. Gli elementi principali di questo modello sono le tabelle e le relazioni. Quest'ultime, sono descritte come delle entità che raggruppa diverse tuple, caratterizzate da degli attributi che contengono dei valori definiti in un dominio di valori predefinito. Nei database relazionali è obbligatorio dover definire a priori lo schema della base di dati, ovvero definire le relazioni, gli attributi e i domini.

Nei DBMS relazionali una transazione è descritta come una sequenza di operazioni eseguite in modo atomico, garantendo la consistenza e l'integrità della base di dati, anche in caso di fallimento. I database relazionali garantiscono che le transazioni rispettino le proprietà identificate con l'acronimo ACID:

- **Atomicità**: una transazione è atomica se nel momento in cui è terminata tutte gli effetti delle sue operazioni sono visibili. In caso di fallimento la

transazione non avrà alcun effetto. Questa proprietà garantisce la consistenza del database, in quanto non si troverà mai in uno stato intermedio.
- **Consistenza**: una transazione è consistente se nel momento in cui è terminata tutti i vincoli d'integrità sono stati soddisfatti e quindi la base di dati sarà sempre in uno stato consistente.
- **Isolamento**: una transazione è isolata se la sua esecuzione non dipende da altre transazioni.
- **Durabilità**: la condizione di durabilità di una transazione prevede che tutte le modifiche apportate nella transazione vengono memorizzate in modo persistente, anche in caso di guasti.

2.2.2 Basi di dati NoSQL

I database NoSQL sono una tipologia di database che non seguono il tradizionale modello relazionale. Esistono diversi tipi di database NoSQL che differiscono sulla tipologia del modello di dati che si vuole memorizzare. Tali database sono una soluzione ideale per tutte quelle applicazioni che necessitano di database con una struttura flessibile, altamente scalabili e con una bassa latenza, in modo da garantire prestazioni elevate. Le caratteristiche principali dei database NoSQL sono le seguenti:

- **Flessibilità**: i database NoSQL non richiedono la definizione di uno schema predefinito. Con l'assenza di tale vincolo si possono memorizzare dati semi-strutturati e non-strutturati. Inoltre, questa caratteristica permette di modificare facilmente e velocemente la struttura dei dati.
- **Scalabilità**: i database NoSQL sono progettati per poter scalare facilmente in modo orizzontale con la presenza di diversi cluster distribuiti di macchine, a cui si attribuiscono a ciascuno di essi dei dati e del carico di lavoro. Con la scalabilità orizzontale si possono aumentare i volumi di memorizzazioni e ottenere prestazioni più elevate semplicemente aggiungendo componenti hardware per la memorizzazione. Questa operazione può risultare più economica rispetto rispetto ad aumentare le capacità computazionali e di memorizzazione di una singola macchina. Il dimensionamento orizzontale garantisce anche la disponibilità di accesso alla base di dati anche in caso di guasti alle singole macchine. L'aspetto negativo della scalabilità orizzontale è l'aumento della complessità dell'architettura del sistema.
- **Prestazioni**: i database NoSQL garantiscono prestazioni elevate con l'utilizzo di schemi di accesso ottimizzati per la tipologia di dato a cui si vuole accedere, in particolare quando si vuole fare accesso a dati altamente connessi tra di loro.

L'uso di sistemi distribuiti garantisce la scalabilità e la resilienza dei dati al netto della consistenza e dell'accuratezza immediata dei dati. Nel 2000 Eric Brewer enunciò il teorema CAP, in cui si afferma che un sistema informatico distribuito possa soddisfare contemporaneamente al massimo due tra le seguenti caratteristiche:

- consistenza: in ogni istante tutti i nodi connessi al sistema distribuito accedono agli stessi dati.
- disponibilità: un client ha la garanzia di poter ricevere una risposta ad ogni richiesta, anche nel caso in cui uno o più nodi fosse temporaneamente non disponibile.
- tolleranza di partizione: garanzia di funzionamento del sistema distribuito anche nel caso di problemi di comunicazione e visibilità tra i nodi.

Il teorema CAP è alla base della nascita del modello BASE, utilizzato dai database NoSQL, come alternativa alle proprietà ACID, offerte dai database relazionali. Il modello BASE garantisce le seguenti proprietà:

- **Basic Availability**: i database NoSQL distribuiscono i dati su diversi sistemi di memorizzazione garantendone la disponibilità anche in caso di diversi malfunzionamenti.
- **Soft State**: lo stato della base di dati può cambiare nel tempo e il modello BASE prevede che le proprietà a consistenza non sia gestita completamente dagli sviluppatori.
- **Eventual Consistency**: nei database NoSQL la consistenza dei dati è garantita ma non è possibile sapere l'istante in cui la base di dati passi in uno stato coerente in seguito a delle modifiche.

I principali modelli dei database NoSQL sono:

Key-Value Nei database di tipo key-value i dati sono memorizzati come coppia chiave-valore in cui il valore può assumere un tipo qualsiasi. È possibile avere come valore anche un ulteriore livello di chiave-valore.

Wide-column I database di tipo wide-column sono caratterizzate da una memorizzazione dei dati in colonne. Per questo motivo, la ricerca di un valore in una determinata colonna è molto veloce. Ogni riga di una tabella può contenere un numero diverso di colonne. Lo svantaggio più grande che si ha con questi tipi di database si verifica nel momento in cui si aggiungono nuovi elementi alla base di dati, poiché il salvataggio richiede l'aggiornamento di tutte le colonne.

Document I database NoSQL documentali sono organizzati in documenti semi-strutturati, contenenti coppie di campi e valori. Una delle caratteristiche migliori è che essi possono scalare orizzontalmente. Tipicamente un documento ha una struttura simile alle entità che rappresentano i dati in una applicazione.

Graph Nei database a grafo, i dati sono memorizzati come nodi e archi. I nodi sono caratterizzati da una o più etichette e possono contenere un numero qualsiasi di attributi detti proprietà, salvati come coppie di chiave-valore. Gli archi offrono una connessione tra due entità e sono definiti da un nome e possono avere un numero arbitrario di proprietà, che solitamente sono proprietà quantitative. Essi hanno sempre una direzione, un nodo iniziale e un nodo finale.

2.2.3 Confronto e valutazioni

Un confronto tra le caratteristiche principali dei database relazionali e quelli NoSQL è descritto in tabella 2.1.

Database relazionali	**Database NoSQL**
Modello relazionale con schema dei dati strutturato	Modello variabile con schema dei dati flessibile
Scalabilità verticale	Scalabilità orizzontale
Proprietà ACID	Proprietà BASE

Tabella 2.1: Confronto tra le caratteristiche di base dei database sequenziali e i database NoSQL

La scelta della tipologia della base dati può essere effettuata in base al tipo di utilizzo che se ne vuole fare. Per esempio i database relazionali sono raccomandati per applicazioni in cui si devono calcolare dei valori a partire da informazioni memorizzate in diverse tabelle, mentre se si vuole descrivere come un utente si muove da un determinato punto ad un altro sarebbe più opportuno utilizzare i database a grafo. Se si volessero memorizzare delle informazioni strutturate allora potrebbe essere necessario utilizzare i database documentali.

2.2.4 Neo4j

Neo4j è un database open source non relazionale sviluppato interamente in Java. Esso appartiene alla tipologia di database a grafo.

Negli ultimi anni i database a grafo stanno diventando sempre più popolari come alternativa ai database relazionali. Un database a grafo è un database NoSQL in cui i dati sono memorizzati sotto forma di nodi e archi, che rappresentano rispettivamente le entità e le relazioni tra di esse.

I database a grafo hanno come punti di forza il fatto di essere altamente scalabili e flessibili. Essi possono contenere i dati senza avere vincoli strutturali. Infatti, a differenza dei database relazionali, che salvano i dati in modo altamente strutturato in tabelle con delle colonne ben definite, i database a grafo non richiedono alcuna struttura pre-definita. Inoltre è possibile modificare la struttura ogni volta che si inserisce un nodo nuovo.

Rispetto ai database relazionali, i database a grafo offrono prestazioni eccellenti quando si devono trattare dati altamente connessi tra loro, indipendentemente dalle dimensioni dei dataset. Questa caratteristica è la conseguenza del fatto che le relazioni vengono salvate insieme ai dati nel modello e non vengono calcolate ogni volta attraverso delle query.

Neo4j ha un proprio linguaggio chiamato Cypher. Esso è molto simile a SQL, con la principale differenza che lavora con i nodi e non con le tabelle. Cypher rappresenta le entità principali dei database a grafo nei seguenti modi:

- i nodi sono rappresentati mediante il costrutto "(X)";
- le relazioni sono rappresentate dal simbolo freccia "->";
- la tipologia di relazione è racchiusa tra due parentesi quadre "[:RELAZIONE]".

Un esempio di rappresentazione di nodi è archi mediante il linguaggio Cypher è il seguente:

(X)-[:RELATION]->(Y).

2.3 Data Science

La Data Science è una disciplina che ha come obiettivo quello di estrarre significato da volumi più o meno grandi di dati. L'intero processo della Data Science può essere decomposto nelle seguenti fasi:

- **Generazione**: la generazione dei dati può avvenire in modo passivo, attivo o automatico. Nella modalità passiva solitamente i dati sono strutturati, come per esempio le transazioni eseguite per gli acquisti online; la modalità attiva prevede la generazione di dati semi-strutturati o non strutturati, per esempio i dati di navigazione su un social network; mentre per la modalità automatica si hanno dati generati da componenti digitali, come ad esempio sensori.
- **Acquisizione**: in seguito alla generazione dei dati è necessario acquisirli in appositi data center e sottoporli a un processo di pre-processing.

- **Memorizzazione**: la memorizzazione dei dati all'interno di una base di dati garantisce la loro persistenza.
- **Analisi**: attraverso dei processi di analisi è possibile estrarre informazioni utili e la conoscenza dai dati.

2.3.1 Big Data

Il termine "Big Data" si riferisce ai dati la cui scala, diversità e complessità richiedono nuove architetture, tecniche, algoritmi e analisi per gestirli ed estrarne valore e il significato nascosto [13].

All'inizio degli anni 2000, Dough Laney espose in un articolo la definizione di Big Data come "le tre V":

- **Volume**: il volume dei dati cresce esponenzialmente negli anni. Tali dati sono generati da sorgenti di diversa natura e spesso non sono strutturati e hanno un valore sconosciuto.
- **Velocità**: molte fonti generano i dati molto velocemente e spesso è richiesto di gestirli in tempo reale.
- **Varietà**: i dati prodotti differiscono nel formato, nel tipo e nella struttura. Essi possono essere dati strutturati o non strutturati, possono essere transazioni, dati di stock, dati prodotti da sensori...

Al modello definito da Laney sono state successivamente aggiunti i termini Veridicità e Valore. I dati ricevuti da una sorgente sono spesso utilizzati per prendere delle decisioni, per questo è necessario capire quanto siano veritieri, assegnandogli indice di affidabilità. Infine, la caratteristica più importante dei big data è quella di poter trasformare i dati in valore.

2.3.2 Processo ETL: Extract, Transform, Load

Extract, Transform, Load (ETL) è un processo di integrazione dei dati che permette l'estrazione di informazioni da diverse sorgenti e la loro successiva elaborazione e salvataggio in un sistema di persistenza. Un processo ETL viene eseguito durante il primo popolamento del data storage e durante gli aggiornamenti periodici dei dati. L'obiettivo principale di un processo ETL in un'azienda è quello di creare una sorgente dati di alta qualità e in uno stato utile per attività di analisi e scopi commerciali. Il processo di ETL è spesso utilizzato per generare data warehouse, replicare database, migrare dati nel cloud e applicare algoritmi di machine learning e intelligenza artificiale.

Estrazione L'estrazione è l'operazione iniziale di un processo ETL e prevede il caricamento dei dati da una o più sorgenti in una staging area. I dati possono provenire da sorgenti eterogenee, ad esempio server SQL o NoSql, sistemi CRM e ERP, registri di attività, file di testo o documenti.

Trasformazione e pulitura Durante il processo di trasformazione i dati grezzi vengono sottoposti a diverse funzioni per poter essere successivamente salvati. La trasformazione è uno step fondamentale del processo ETL perché permette di garantire la qualità dei dati che saranno salvati nel sistema di persistenza. Tra le fasi di estrazione e caricamento vengono eseguite le operazioni di pulitura, filtraggio, deduplicazione, formattazione e joining dei dati. Queste operazioni sono fondamentali in un processo ETL perché permettono di garantire la qualità dei dati che saranno salvati nel sistema di archiviazione.

Caricamento L'operazione conclusiva di un processo ETL è il caricamento, in cui avviene il trasferimento dei dati elaborati dalla staging area ad un database, un datastore o un data warehouse.

2.3.3 Data Mining

Il termine data mining si riferisce all'insieme di tecniche e metodologie che hanno lo scopo di estrarre automaticamente, tramite l'uso di algoritmi, di informazioni dai dati [14]. L'obiettivo è quello di ricavare un significato non noto e utile ai fine applicativi.

Il data mining può essere di tipo predittivo o descrittivo. Nel primo caso l'obiettivo è quello di creare un modello che sia in grado di determinare un valore in output di una variabile che dipenda da un insieme di fattori e proprietà; nel caso in cui l'output fosse di tipo numerico si parla di modello di regressione, mentre nel caso di output nominale si ha un modello di classificazione. Nell'altro caso, un modello descrittivo, ha come obiettivo quello di ricavare delle regole derivanti dai dati, attraverso delle associazioni tra le proprietà e gli attributi dei record, oppure raggruppando cluster di elementi con proprietà simili.

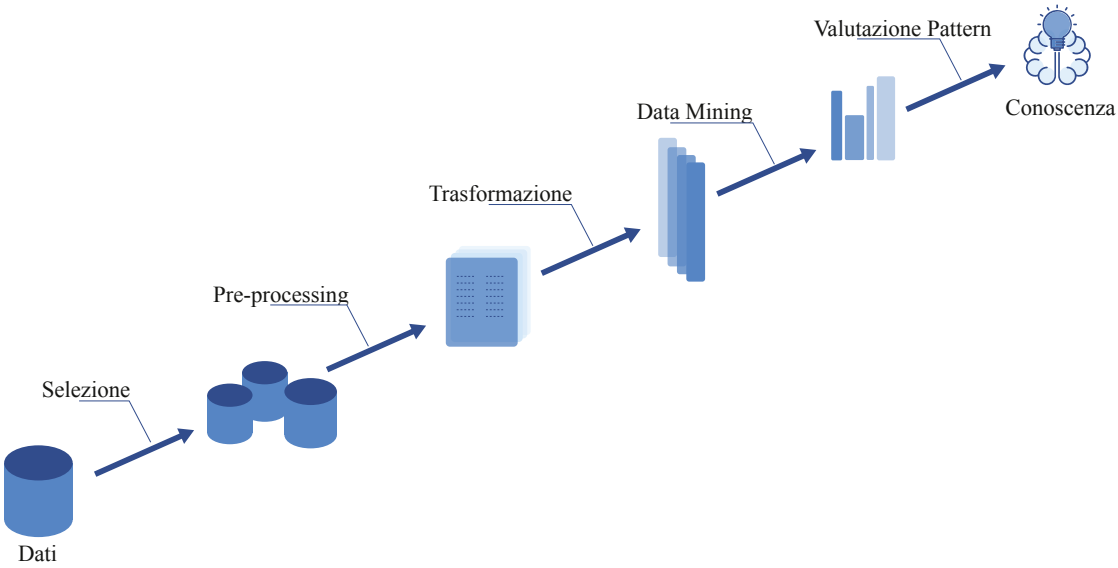

Figura 2.4: Data mining - Knowledge Discovery from Data

Con il termine Knowledge Discovery from Data (KDD) si indica l'intero processo di estrazione della conoscenza dai dati (immagine 2.4). Tale processo si pone come obiettivo quello di ricercare modelli comprensibili per l'interpretazione dei dati. Il punto di partenza è dato dai dati provenienti dal mondo reale. Il processo è suddiviso in 5 fasi:

1. **Selezione**: il primo passo è quello di selezionare i dati utili per l'analisi che si vuole effettuare. La selezione può avvenire in orizzontale, filtrando le tuple in base a delle condizioni, oppure in verticale, selezionando le feature.
2. **Pre-processing**: successivamente alla selezione, avviene la fase di pre-processing, fondamentale per poter estrarre dei pattern di valore. Durante questa fase possono avvenire le operazioni di cleaning dei dati, gestione dei dati mancanti, feature engineering...
3. **Trasformazione**: la trasformazione raggruppa l'insieme di operazioni per la generazione del dataset finale da utilizzare nell'algoritmo di data mining. Tra le operazioni principali ci sono quelle di riduzione della dimensionalità, attraverso il processo di feature selection.
4. **Data Mining**: fase in cui si utilizza uno o più algoritmi di data mining per poter ricavare l'informazione intrinseca nei dati.
5. **Interpretazione**: l'interpretazione è la fase conclusiva, in cui i risultati sono valutati e interpretati da una persona nel dominio di analisi. Per interpretare dei modelli talvolta è necessario doverli rappresentare graficamente, se possibile.

Le tecniche di analisi utilizzate nel data mining sono le regole di associazione, la

classificazione e il clustering [14].

2.3.4 Regole di associazione

Le regole di associazione permettono di estrarre pattern frequenti da un dataset transazionale [15].

TID	Elementi
T0	{A, B, C, D, E}
T1	{A, B, C}
T2	{C}
T3	{A, B, D, E}
T4	{B, E}
T5	{A, B, E}

Tabella 2.2: Esempio di un dataset transazionale per l'estrazione di regole di associazione

Osservando la tabella 2.2, in cui sono rappresentate 6 transazioni con le relative liste di elementi presenti in ciascuna di esse, si può definire la seguente regola di associazione:

A, B $\rightarrow$ C

in cui A e B sono il corpo della regola e C è detta testa della regola.

Si definisce *itemset* un insieme di uno o più item e *k-itemset* un itemset con k item. Il *supporto* è il rapporto tra il numero di transazioni che contengono un itemset e il numero totale di transazioni. Gli *itemset frequenti* sono quegli itemset che hanno il supporto maggiore di un valore di soglia detto supporto minimo (minsup). La *confidenza* è la probabilità condizionale di trovare un item (A) avendo già trovato un altro item (B).

L'estrazione delle regole di associazione, a partire da un insieme di transazioni, avviene nel momento in cui si soddisfano determinati vincoli, tra cui ad esempio:

- supporto $\geq$ minsup
- confidenza $\geq$ minconf

Le regole di associazioni ricavate sono **complete** se tutte soddisfano entrambi i vincoli, e si definiscono **corrette** se solo le regole soddisfano i vincoli.

2.3.5 Classificazione

Dato un dataset contenente un insieme di etichette di classe (Y) e un insieme di oggetti contrassegnati da un'etichetta di classe (X), è possibile definire un modello in grado di assegnare l'etichetta appropriata agli oggetti senza etichetta [16].
Applicazioni tipiche della classificazione sono per esempio il riconoscimento dell'intenzione dei clienti di eseguire un'azione, il riconoscimento di una frode, ecc...

Nei problemi di classificazione è necessario partizionare il dataset di partenza in training set e test set. La prima partizione contiene i dati su cui il modello viene costruito, mentre la seconda è utilizzata per valutare la bontà del modello generato. La dimensione delle due partizioni non è definibile a priori, ma è possibile provare diverse dimensioni e valutare le rispettive performance per poter trovare i valori ottimali.

Di seguito sono descritte due delle tecniche di classificazione maggiormente utilizzate:

Decision Tree Un albero di decisione è un grafo utilizzato per poter dedurre l'etichetta di classe di un nodo a partire dai suoi attributi. In un albero di decisione sono presenti molteplici nodi in cui si effettua un test su un determinato attributo e diversi rami uscenti dai nodi. La scelta del ramo da percorrere dipende dal valore dell'attributo dell'elemento preso in considerazione. I nodi foglia assegnano una classificazione. Il nodo radice è il primo nodo che si deve percorrere per poter classificare un'istanza.
Esistono diversi algoritmi per la generazione degli alberi decisionali, tra cui: l'algoritmo di Hunt, CART, ID3, ecc... La scelta del miglior attributo di split avviene valutando la misura di impurità del nodo, che può essere misurata in diversi modi, per esempio:

- **Gini index**: dato un nodo t, l'indice Gini è definito come

$$GINI(t) = 1 - \sum_j [p(j|t)^2]$$

in cui p(j|t) è la frequenza relativa della classe j nel nodo t.
Diramando un nodo, la qualità di split è calcolata come:

$$GINI_{split} = 1 - \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

in cui n_i è il numero di record nel figlio n e n è il numero totale di record nel nodo p.

- **Entropia**: l'entropia nel nodo t è definita come

$$Entropy(t) = -\sum_j p(j|t) log_2 p(j|t)$$

in cui p(j|t) è la frequenza relativa della classe j nel nodo t.
Essa è massima quando tutti i record sono distribuiti equamente in tutte le classe (grado di impurità elevato), ed è minima quando tutti i record appartengono ad una sola classe (grado di impurità basso).

Random Forest la foresta casuale è una tecnica di apprendimento d'insieme, in cui diversi modelli sono combinati per poter aumentare la stabilità e l'accuratezza e per evitare il problema di overfitting [16]. Gli algoritmi di random forest sono definiti come un insieme di alberi decisionali e la classe viene assegnata effettuando un voto di maggioranza (tra tutti gli alberi disponibili). Rispetto agli alberi decisionali essi hanno una maggiore accuratezza ma presentano anche un tempo di creazione del modello maggiore, tuttavia nel complesso la generazione del modello è molto veloce, così come anche il suo utilizzo per poter effettuare delle classificazioni.

Altre tecniche di classificazione sono: Naïve Bayes, k-Nearest Neighbours (k-NN), Support Vector Machines (SVM), ecc...

Gli algoritmi di classificazione si basano su un numero variabile di parametri, chiamati iperparametri, che possono essere impostati in fase di sviluppo. Tali parametri sono specifici per ogni algoritmo di classificazione e non c'è un set predefinito di valori ottimali. La scelta degli iperparametri può risultare molto complessa, per questo esistono diverse tecniche tra cui:

- **Grid search**: selezione dei valori degli iperparametri eseguendo la validazione incrociata su un'intera griglia di possibili parametri definita a priori. Si esplorano completamente tutte le possibili combinazioni.
- **Random search**: selezione dei valori selezionando un numero ridotto di combinazioni da una griglia di possibili valori. Questa tecnica permette di ricercare i migliori parametri in uno spazio di ricerca maggiore.

2.3.6 Metriche di valutazione

I modelli di Machine Learning vengono valutati su un insieme di dati di test ed è importante avere delle metriche per poter misurare le prestazioni e capire se il modello è accettabile o se necessita di miglioramenti. Esistono diverse metriche,

tra cui l'**accuratezza**, che è una delle più semplici e più utilizzate. Essa indica il rapporto tra il numero di previsioni corrette e il numero totale di previsioni:

$$Accuratezza = \frac{\#\text{ previsioni corrette}}{\#\text{ previsioni totali}}$$

Tuttavia ci sono casi in cui la precisione non è sufficiente, come per esempio nel caso in cui si utilizzi un dataset sbilanciato verso una classe, in quanto qualsiasi modello che classificasse i record con l'etichetta di classe predominante otterrebbe una precisione molto vicina al 100%. Perciò in alcuni casi è necessario utilizzare delle ulteriori metriche che siano in grado di differenziare le valutazioni per tutte le classi. Infatti esistono metriche che si possono utilizzare per valutare singolarmente tutte le classi e sono il richiamo e la precisione.

Il **richiamo** è definito come:

$$Richiamo = \frac{\#\text{ oggetti correttamente assegnati alla classe C}}{\#\text{ oggetti appartenenti alla classe C}}$$

La **precisione** invece è ricavabile con il seguente rapporto:

$$Precisione = \frac{\#\text{ oggetti correttamente assegnati alla classe C}}{\#\text{ oggetti assegnati alla classe C}}$$

Infine la misura F1... Dato che il richiamo e la precisione non si possono massimizzare contemporaneamente, si può utilizzare una nuova misura chiamata **F1 score**, in cui si esegue una media armonica delle due misure:

$$F1score = 2 \cdot \frac{Precisione \cdot Richiamo}{Precisione + Richiamo}$$

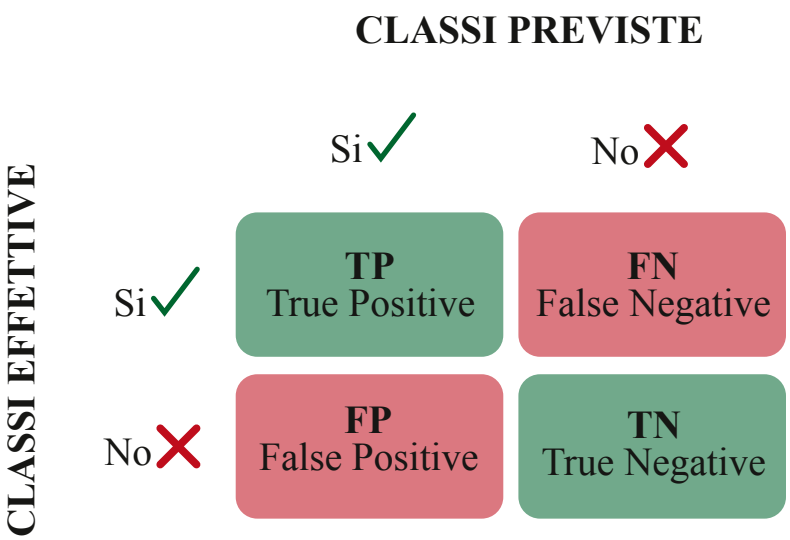

Figura 2.5: Matrice di confusione

Si definisce **matrice di confusione** una matrice composta da righe in cui ci sono le classi predette del test set e da colonne in cui ci sono le classi reali del test set. Una rappresentazione generica di una matrice di confusione è riportata in figura 2.5. Ogni cella della matrice contiene il numero di campioni classificati con la classe prevista e la sua classe effettiva. La diagonale contiene le predizioni corrette, mentre gli altri due elementi contengono le classificazioni errate.

Utilizzando la matrice di confusione è possibile ridefinire le metriche viste precedentemente:

$$Accuratezza = \frac{TP + TN}{TP + TN + FP + FN}$$

Per la classe positiva si avrebbero le seguenti misure di richiamo e precisione:

$$Richiamo = \frac{TP}{TP + FN}$$

$$Precisione = \frac{TP}{TP + FP}$$

Capitolo 3

Metodologia

Questo capitolo illustra la metodologia adottata per il progetto di tesi, dall'acquisizione dei dati alla generazione del modello predittivo. Nella sezione 3.1 si illustrano le fasi del progetto. Successivamente la 3.2 offre una panoramica sulla struttura del sito web considerato, utile per poter classificare correttamente le pagine visitate dai clienti.

3.1 Approccio

La figura 3.1 rappresenta gli strati esterni del progetto di tesi. Questa rappresentazione mette in risalto il contesto e l'obiettivo della tesi, mostrando gli elementi esterni che agiscono e influenzano il risultato finale. In ingresso ci sono i dati che i clienti generano durante il loro utilizzo sul sito web dell'e-commerce B2B, dalla navigazione tra le categorie all'acquisto di prodotti o all'abbandono del sito. Un software di tracking raccoglie tali dati e li rende accessibili al sistema che li elabora e produce in uscita un modello predittivo, che può essere utilizzato per poter mostrare contenuti dinamici e personalizzati in base al tipo di cliente.

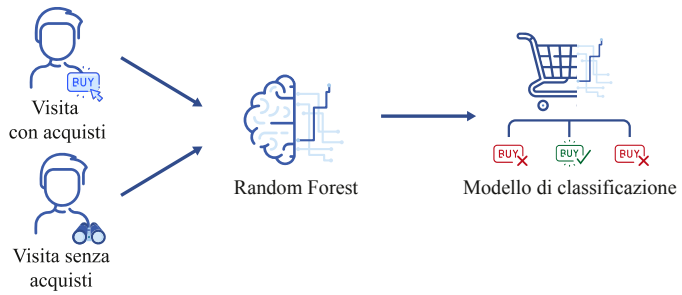

Figura 3.1: Input e output del sistema da implementare

Entrando nel dettaglio del sistema (figura 3.2) si osserva la presenza di due componenti fondamentali: un database e un sistema di machine learning.

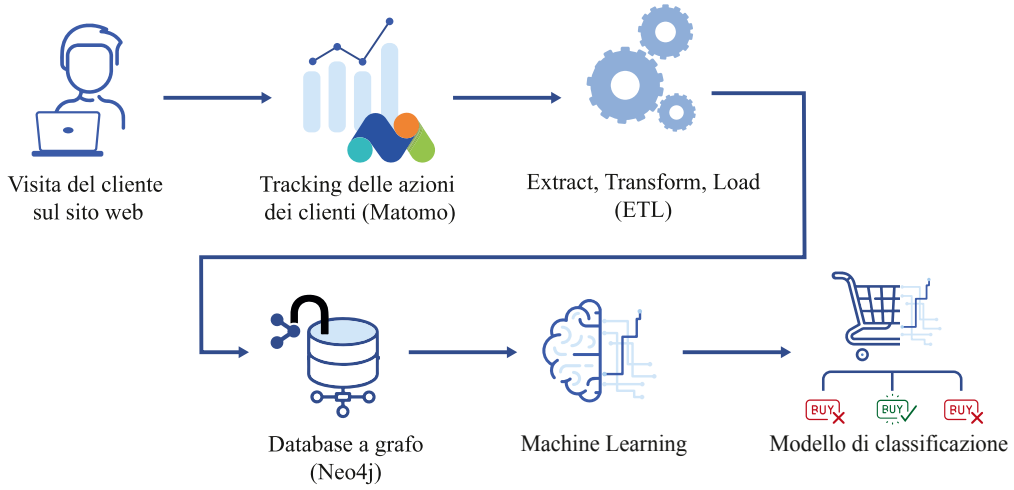

Figura 3.2: Metodologia adottata

Il punto di partenza di questo progetto corrisponde con l'implementazione di un algoritmo, basato sul processo "Extract, Transform, Load" (ETL), per poter memorizzare periodicamente l'enorme quantità di dati provenienti da un software di tracking, con una particolare attenzione all'efficienza e la correttezza dei dati. Per addestrare correttamente un modello predittivo, i dati storici devono soddisfare standard di qualità elevati: i dati devono essere corretti, de-duplicati ma allo stesso tempo bisogna considerare solo i dati effettivamente utili per la generazione del modello, escludendo per esempio quelli affetti da bias [17].

La base dati scelta è Neo4j, un database non relazionale a grafo, su cui si effettua un'analisi per la generazione di un modello predittivo. La rappresentazione a grafo si presta particolarmente bene nella memorizzazione e l'analisi delle sessioni di navigazione dei clienti su un sito web e-commerce: tra le caratteristiche fondamentali di un database a grafo c'è la flessibilità e la semplicità con cui si possono salvare le relazioni; in particolare, per ogni sessione di un cliente, è possibile salvare le azioni in modo sequenziale, in modo che ognuna di esse abbia una proprietà con la precedente del tipo "next".

Il progetto procede successivamente con la ricerca di pattern comportamentali degli utenti, attraverso un algoritmo di machine learning in Python. L'obiettivo di questa fase è quello di ottenere un modello predittivo per poter classificare le sessioni dei clienti, utilizzando direttamente i dati memorizzati all'interno del

database a grafo.

Il dataset, prima di essere utilizzato negli algoritmi di machine learning, viene sottoposto a una fase di *preprocessing*, con una gestione dei valori mancanti, la rimozione del rumore nei dati e la gestione di eventuali inconsistenze.
Successivamente, per poter analizzare i dati ed ottenere il modello predittivo, si esegue una fase di selezione delle caratteristiche (*feature selection*) combinata con l'estrazione delle caratteristiche (*feature extraction*) per poter rispettivamente selezionare solo le informazioni significative per addestrare il sistema e per aggiungere informazioni utili a partire dai dati esistenti.
Infine, in base delle caratteristiche selezionate, il sistema viene addestrato per generare il modello predittivo.

Il progetto è stato sviluppato interamente in Python. Per poter interrogare e manipolare il database a grafo si è utilizzato il linguaggio Cypher. Il software di tracking utilizzato dall'azienda si chiama Matomo e mette a disposizione un'API per poter richiedere le visite dei clienti.

3.2 Struttura del sito web

La struttura del sito web dell'e-commerce considerato per questo progetto di tesi è stata definita analizzando gli URL (URL) visitati dai clienti durante la navigazione. Durante l'intero periodo del progetto il sito non ha subito modifiche strutturali.

Una risorsa web è identificata univocamente da una sequenza di stringhe detta URI (Universal Resource Identifier). Un Uniform Resource Locator è un tipo specifico di URI ed è utilizzato per identificare una risorsa in base alla sua locazione su Internet. Un URL è comunemente strutturato in quattro componenti [18]:

- **Schema** - specifica il protocollo utilizzato per accedere alla risorsa, per esempio HTTP/HTTPS.
- **Host** - il nome dell'host identifica l'host che possiede la risorsa. Il nome dell'host è seguito dal numero di porta, che può essere omessa nel caso in cui un server offra dei servizi su dei numeri di porta noti.
- **Path** - il percorso localizza in modo assoluto la risorsa memorizzata nel filesystem dell'host. Un percorso può essere organizzato in segmenti intermedi, separati da un singolo carattere slash "/".
- **Query string** - una stringa di query è un insieme di coppie nome-valore, separate dal carattere "&", posta al termine del percorso per fornire informazioni alla risorsa richiesta, sotto forma di parametri.

Il numero di azioni totali, associate alle visite online dei clienti dell'e-commerce nel periodo che va dal 23 giugno 2021 al 30 giugno 2021, di poco superiore a 210 mila, è sufficiente per poter dedurre la struttura del sito web. Quasi tutte le azioni tracciate dal software di tracking sono associate a una determinata pagina del sito web, raggiungibile al corrispondente URL. Nel periodo considerato, il numero totale di URL è di 210.984. Escludendo i duplicati si ottengono 29.879 URL unici. Rimuovendo le query string il numero si riduce a 15.365.

Nonostante le semplificazioni applicate, la quantità di URL è elevato e non permette di ricavare facilmente la struttura del sito osservandoli singolarmente. Una possibile soluzione è quella di dividere ciascun URL, sfruttando la presenza del carattere "/", in modo da ottenere tutti gli elementi che compongono il percorso della locazione della pagina web sull'host e memorizzarli in una struttura gerarchica ad albero. Quindi, scegliendo opportunamente il livello di profondità, che rappresenta il livello di dettaglio che si vuole ottenere, si può stampare ricorsivamente la struttura memorizzata (figura 3.3). Le pagine principali e i relativi URL sono riassunti in figura 3.3.

```
https://www.brandsdistribution.com
└─ /[codice stato]
   ├─ /                                  Home
   ├─ /catalog                           Catalogo
   │  ├─ /                               //
   │  ├─ /2                              //
   │  ├─ ⋮                               //
   │  └─ /designer                       //
   │     └─ /[nome designer]             //
   │        └─ /[categoria]              //
   │           └─ /[sub-categoria]       //
   │              └─ /[nome prodotto]     Prodotto
   ├─ /login                             Login
   ├─ /subscribe/...                     Iscrizione
   ├─ /profile
   │  ├─ /data                           Dati profilo
   │  ├─ /dispatch                       //
   │  ├─ /invoice                        //
   │  ├─ /orders/[index]                 Post-acquisto
   │  └─ /returnrequests/[index]         //
   ├─ /cart                              Carrello
   ├─ /checkout                          Checkout
   │  ├─ /payment
   │  ├─ /done
   │  ├─ /shipping
   │  ├─ /billing
   │  └─ /cancelled
   ├─ /designers                         Lista designer
   └─ /cms                               Informazioni
      ├─ /ph-[nome brand]
      ├─ /size-table
      ├─ /about-us
      ├─ /faq
      └─ ⋮
```

Figura 3.3: Struttura del sito web

Capitolo 4

Salvataggio delle visite in una struttura a grafo

Questo capitolo descrive la fase iniziale del progetto di tesi, ovvero quella del trasferimento dei dati delle visite dei clienti in una struttura dati a grafo. La struttura del capitolo segue principalmente le tre fasi del processo ETL ed ha una sezione iniziale (sez. 4.1) dedicata alla progettazione della base dati e una finale (sez. 4.5) per la valutazione del processo in termini di prestazioni.

4.1 Progettazione della base dati

4.1.1 Struttura del grafo

La tabella 4.1 mostra una panoramica dei nodi più significativi e le loro proprietà. Le proprietà sottolineate identificano le chiavi primarie. I nodi sono legati tra loro tramite delle semplici relazioni senza alcuna proprietà, per esempio il nodo *Customer* è legato al nodo *Visit* tramite la relazione PAYS_A. I nodi delle azioni hanno una relazione etichettata come :NEXT per tenere traccia dell'ordine temporale in cui sono avvenute. L'elenco completo dei nodi e delle relazioni sarà mostrato nell'ultima sezione 4.5. Si noti inoltre che il nodo *Action* richiede una certa flessibilità in quanto in base al tipo di azione può contenere o meno alcune proprietà (quelle con l'asterisco). Per esempio le proprietà event e quantity sono caratteristiche delle azioni di tipo "cartUpdate".

Label	**Proprietà**
(:Visit)	visitId serverDateTime localHour duration converted referrerType
(:Action)	actionId type dateTime timeSpent siteSearchKeyword event quantity url
(:Page)	pageId url pageType
(:Order)	orderId dateTime revenue items

Tabella 4.1: Label e proprietà dei principali nodi del database Neo4j

4.1.2 Definizione di vincoli di integrità e indici

I vincoli di integrità sono dei predicati, applicati a un determinato nodo, che servono per perseverare l'integrità della base dati. I vincoli sono delle regole di dominio, usati comunemente per garantire l'univocità dei nodi, in base a una o più proprietà. In Neo4j l'aggiunta del vincolo d'integrità crea in automatico un indice a quella proprietà.

Un indice è una struttura dati che permette di ridurre i tempi di accesso ai dati del database in quanto la ricerca si riduce a un sottoinsieme di valori da leggere. Gli indici vengono utilizzati nei database a grafo come punti di partenza

di attraversamento della struttura dati per poter ridurre i tempi di ricerca e quindi migliorare le prestazioni. Un indice composto è un indice applicato su più proprietà di un certo nodo.

Per il progetto di tesi sono stati creati i vincoli sulle chiavi primarie di tutte le tabelle e in modo implicito è avvenuta la generazione degli indici sui medesimi campi. Diversamente, un indice alla proprietà webpageId del nodo *Page* sarebbe stato necessario per poter garantire delle prestazioni elevate per l'inserimento nel database (si veda il codice 4.4.1).

4.2 Estrazione dei dati

Con l'utilizzo della libreria python *requests* viene fatta una richiesta HTTP all'API Matomo [19] per ottenere le informazioni riguardanti le visite dei clienti in un determinato periodo, specificando i seguenti parametri:

- **method**: l'API mette a disposizione diverse funzionalità; per richiedere i dettagli delle visite dei clienti si utilizza il metodo *Live.getLastVisitsDetails*.
- **idSite**: id del sito web di interesse.
- **period**: periodo per il quale si richiedono le statistiche; i valori accettati sono: *day, week, month, year e range*.
- **date**: data o intervallo di date nel caso in cui il periodo specificato è di tipo *range*; il formato delle date è *yyyy-mm-dd*.
- **format**: formato dell'output, ad esempio *json, xml, csv....*
- **filter_limit**: quantità di risultati nella risposta; per avere una risposta completa si usa il valore *-1*.
- **token_auth**: token usato per l'autenticazione all'API.

La libreria pandas [20] fornisce delle strutture e degli strumenti, nel linguaggio Python, per analizzare e manipolare i dati. Una delle funzionalità che offre è quella di memorizzare i dati in input, ottenute nel formato JSON, in una struttura tabulare bidimensionale chiamata DataFrame. La libreria mette a disposizione diverse funzioni e attributi da applicare al DataFrame per poter analizzarne il contenuto, tra cui:

- **shape**: attributo che memorizza il numero di righe e colonne come tuple(#rows, #columns)
- **head(n)**: metodo che permette di visualizzare le prime n righe del DataFrame.
- **info()**: metodo che mostra diverse informazioni sul DataFrame, tra cui il numero di righe e colonne, la memoria utilizzata, il tipo di dato di ogni colonna e il numero di elementi non nulli (NaN).

Comprendere la struttura dei dati di partenza è il primo passo fondamentale per poter procedere ad implementare correttamente le successive fasi del progetto. Nel dettaglio, a seguito di una richiesta HTTP all'API, ciò che si ottiene è una lista di oggetti json, che rappresentano le singole sessioni dei visitatori sul sito web considerato. I dati ottenuti vengono salvati all'interno di un DataFrame con il metodo json_normalize(data), preparati ed infine memorizzati nel database.

Come mostrato in tabella A.1, una visita contiene 103 attributi, alcuni dei quali sono superflui, in quanto ci sono dei dati mancanti, oppure perché sono dati ripetuti con formati diversi (per esempio il tempo impiegato a eseguire un'azione è presente come "timeSpent" e "timeSpentPretty", oppure "lastActionTimestamp" e "lastActionDateTime"). Ad ogni visita è associato un numero arbitrario di azioni, memorizzato nell'attributo "actionDetails", ognuna delle quali è caratterizzata, in base al suo tipo, a un numero variabile di attributi (una visuale complessiva dei 36 possibili parametri è mostrata in tabella A.2.

4.3 Preparazione dei dati

Nella tabella 4.2 sono rappresentati i dettagli degli attributi più significativi delle visite dei clienti. Questa tabella è stata costruita a partire dalla tabella A.1, effettuando una riduzione nella dimensione delle caratteristiche e descrivendo in modo dettagliato ciascun parametro.

Nome	**Tipo**	**Null**	**Descrizione**
idSite	string		Identificativo del sito.
siteName	string		Nome del sito.
idVisit	string		Identificativo della visita.
userId	string	X	Identificativo del cliente; nullo se il cliente non ha effettuato il login.
actionDetails	array		Array contenente le azioni.
firstActionTimestamp	int		Timestamp della prima azione eseguita.
visitLocalHour	string		Ora locale della visita.
visitDuration	string		Durata della visita in secondi.

Continua nella prossima pagina

Tabella 4.2 - continua dalla pagina precedente

Nome	Tipo	Null	Descrizione
referrerType	string		Tipo di referrer, esempi: “direct”, “search”, “campaign”, “website”, “social”...
deviceType	string		Informazioni del device.
deviceBrand	string		Marca del device.
resolution	string		Risoluzione del device.
operatingSystemCode	string		Codice del sistema operativo.
operatingSystemName	string		Nome del sistema operativo.
operatingSystemVersion	string		Versione del sistema operativo.
browserCode	string		Codice del browser.
browserName	string		Nome del browser.
browserVersion	string		Versione del browser.
continent	string		Continente della visita.
country	string		Paese della visita.
city	string		Città della visita.

Tabella 4.2: Attributi selezionati delle visite dei clienti (tabella completa A.1)

In modo analogo a ciò che è stato fatto con le visite, sono state selezionate e riportate in tabella 4.3 un insieme di caratteristiche anche per il DataFrame delle azioni, associate a ciascuna visita.

Nome	Tipo	Null	Descrizione
type	string		Valori possibili: action, event, search, outlink, ecommerceOrder, download.
url	string	X	Nullo se type = “ecommerceOrder” o “ecommerceAbandonedCart”.
pageId	string	X	Identificativo dell’azione. Nullo nel caso di “ecommerceOrder” e “ecommerceAbandonedCart”. Nel caso in cui type = “ecommerceOrder” bisogna considerare come id il campo orderId.
pageIdAction	string	X	Identificativo della pagina web visualizzata (type = “action”)

Continua nella prossima pagina

Tabella 4.3 - continua dalla pagina precedente

Nome	Tipo	Null	Descrizione
timestamp	int		Timestamp dell'azione.
timeSpent	string	X	Tempo impiegato nella pagina quando type = "action". È nullo per l'ultima azione eseguita in una visita.
siteSearchKeyword	string	X	Valore presente solo se type = "search", indica la keyword di ricerca.
eventCategory	string	X	Valore presente se type = "event", la categoria può essere solo di tipo "CartUpdate".
eventAction	string	X	Quando eventCategory non è nulla, eventAction specifica se l'aggiornamento del carrello corrisponde a una aggiunta "add" o una rimozione "remove" di un prodotto.
eventValue	float	X	Quantità di prodotti aggiunti/rimossi durante un evento di "cartUpdate" di un determinato prodotto.
orderId	string	X	Id dell'ordine nel caso in cui type = "ecommerceOrder".
revenueSubTotal	string	X	Subtotale dell'ordine (type = "ecommerceOrder").
items	string	X	Numero di prodotti in un ordine (type è "ecommerceOrder" o "ecommerceAbandonedCart").

Tabella 4.3: Attributi selezionati delle azioni effettuate in una visita (tabella completa A.2)

Ciò che si osserva è che questo dataset, rispetto a quello delle visite, contiene potenzialmente molti valori mancanti, avendo diversi campi che possono essere nulli, che vanno gestiti accuratamente. Inoltre, il dataset delle azioni è anche più complesso perché contiene informazioni che variano in base alla tipologia dell'azione, specificata nel campo "type" e che può assumere uno dei seguenti valori:

- **action** - visita di una pagina (home, categoria, prodotto...);
- **search** - ricerca di un prodotto o una tipologia di prodotto;
- **event** - eventi scaturiti da un'azione del cliente, per esempio la modifica del carrello con l'aggiunta o la rimozione dei prodotti;
- **ecommerceOrder** - completamento dell'ordine;

- **ecommerceAbandonedCart** - azione che si presenta in seguito all'ultimo evento di modifica del carrello se la sessione termina senza alcun acquisto;
- **outlink** - accesso ad un link esterno;
- **download** - scaricamento di una risorsa.

4.3.1 Preparazione delle visite

Questa fase di pulizia del DataFrame permette di ottenere un DataFrame ridotto, in termini di numero di righe e colonne, applicando le funzioni di filtraggio e rimozione delle colonne non utilizzate. Le seguenti operazione rimuovono le visite che hanno durata pari a zero secondi e rimuovono le colonne non utilizzate, facendo riferimento alla tabella 4.2 definita precedentemente.

```
df = df[df['visitDuration'].astype(int) > 0]
df = df[USEFUL_VISIT_FIELDS]
```

Codice 4.1: Filtraggio e selezione delle colonne delle visite

La costante USEFUL_VISIT_FIELDS è un set pre-definito, contente i nomi degli attributi selezionati e specificati in tabella 4.2.

4.3.2 Preparazione delle azioni

Questa fase prevede la trasformazione delle azioni eseguite in ogni visita. Nel codice 4.2 è descritta l'operazione di mapping della colonna actionDetails del DataFrame delle visite, cha ha l'obiettivo di preparare le liste delle azioni, associate a ciascuna visita, per poter essere salvate correttamente nel database.

```
df['actionDetails'] = df['actionDetails'].map(get_actions)
...

def get_actions(data):
    df = pd.DataFrame(data)
    USELESS_ACTION_FIELDS = set(set(df.columns.values) -
                                USEFUL_ACTION_FIELDS)
    df = df.drop(USELESS_ACTION_FIELDS,
                 axis=1,
                 errors='ignore')

    # filter
    df = df[df['type'] != 'ecommerceAbandonedCart']

    # change type for ''event'' types
    df.loc[df['type'] == 'event', 'type'] = 'cartUpdate'
```

```
17
18      # rename ``action'' types
19      df.loc[df['type'] == 'action', 'type'] = 'view'
20
21      # Sort by timestamp
22      df.sort_values(by=['timestamp'], inplace=True)
23
24      # URL mapping
25      df['pageType'] = df['url']
26                          .replace(to_replace=REGEX_DICT, regex=True)
27      # if no regex matched, the pageType will contain an URL
28      # map any URL (that contains '/') with None
29      df['pageType'] = df['pageType']
30                          .replace(r'.*/.*', None, regex=True)
31
32      # Transform back to json object
33      return df.to_dict(orient='records')
```

Codice 4.2: Preparazione delle azioni

La funzione get_actions riceve come parametro la lista di oggetti json, corrispondente alle azioni. Tale lista viene caricata in un DataFrame su cui verranno eseguite delle operazioni in modo simile al DataFrame delle visite visto precedentemente.

I DataFrame generati a partire dalle liste delle azioni contengono informazioni potenzialmente eterogenee, e perciò non hanno sempre una struttura ben definita. Per esempio un DataFrame contenente azioni di tipo "ecommerceOrder" e di tipo "action" avrà una struttura diversa da un DataFrame contenente solo azioni di tipo "action". Per questo motivo, non è possibile definire un insieme di colonne da selezionare, come è stato fatto per il DataFrame delle visite, ma è necessario per prima cosa costruire un insieme di colonne non utili e successivamente rimuovere queste colonne dal DataFrame. Le colonne da scartare sono ricavate effettuando un'operazione di sottrazione tra tutte le colonne del DataFrame e una lista costante di colonne, USEFUL_ACTION_FIELDS, costruita a partire dalla tabella 4.3.
Tra le azioni disponibili nel DataFrame, vengono rimosse le azioni di tipo "ecommerceAbandonedCart" in quanto si tratta di un evento che si presenta in una posizione non sempre definita ed è inoltre una caratteristica della visita che si può ricavare direttamente osservando se tra le azioni di una visita ci sono stati ordini. La colonna type descrive il tipo di azione. Siccome l'evento di tipo "event" descrive solo l'aggiornamento del carrello, si è preferito assegnare un nome più specifico all'evento (cartUpdate).

Prima di trasformare nuovamente il DataFrame in un oggetto dict, si effettua la classificazione delle pagine visualizzate. Per fare ciò si utilizzano delle regole di mapping per assegnare un nome agli URL presenti nelle azioni. Le regole di

classificazione degli URL si basano su espressioni regolari, salvate nella variabile REGEX_DICT e riassunte nella tabella 4.4. Questa tabella è stata costruita osservando la struttura del sito web, descritta nella sezione 3.2.

Nome Pagina	**RegEx**
Home	.*/[a-z]{2}/?$
Prodotto	.*/catalog(/[^/]+){5}/?$
Catalogo	.*/catalog([/?][^/]+){0,4}/?$
Carrello	.*/cart/?$
Checkout	.*/checkout(/([a-z]+/?)?)?$
Login	.*/login.*
Iscrizione	.*/subscribe.*
Dati-profilo	.*/profile((/data)|(/dispatch)|(/invoice))/?$
Post-acquisto	.*/profile((/orders)|(/returnrequests))(/[0-9]*)?$
Lista-brand	.*/designers/?$
Informazioni	.*/cms.*

Tabella 4.4: Regole di mapping per la classificazione delle pagine web tramite espressioni regolari

4.4 Caricamento dei dati

Dopo aver estratto e trasformato i dati si può passare alla fase di caricamento del processo ETL. Il salvataggio avviene nel database a grafo Neo4j avendo come input il DataFrame, convertito in un oggetto dict dopo aver seguito il processo di preparazione descritto nelle sezioni precedenti.

Per ottenere delle prestazioni di caricamento migliori, si procede con un inserimento nel database a blocchi (codice 4.3). Con questo metodo la query per l'inserimento nel database riceve un sottoinsieme dei dati delle visite, evitando di creare una nuova sessione di inserimento per ogni visita. È importante prestare particolare attenzione alla dimensione che si attribuisce ai singoli blocchi per evitare problemi di memoria. La scelta ottimale della dimensione dei blocchi può essere fatta analizzando i tempi di inserimento variando la dimensione e sarà oggetto di discussione dell'ultima sezione di questo capitolo (sez. 4.5).

L'aspetto negativo di questo approccio è un aumento significativo della complessità nella scrittura della query nel linguaggio cypher.

```
def insert_rows(self, query, rows, batch_size=1000):
    # Function to handle insert in batch mode.
    total = 0
    batch = 0
    start = time.time()
    result = None

    while batch * batch_size < len(rows):
        res = list(session.run(query, parameters={
            'rows': rows[batch*batch_size: (batch + 1)*batch_size]
        }))
        total += res[0]['total']
        batch += 1
        result = {``total'': total,
                  ``batches'': batch,
                  ``time'': time.time() - start}
        print(result)

    return result
```

Codice 4.3: Inserimento a blocchi

4.4.1 Cypher Query

La query 4.4.1 riceve come parametro la lista di oggetti dict di visite dei clienti, appositamente preparata per essere inserita nel database. Per ogni visita vengono creati i nodi Website, Visit, Customer, Location e Device. Le informazioni sul device (Brand, Resolution, OS, Browser) sono memorizzate in nodi che hanno una relazione con il nodo Device.

Durante la creazione del nodo Customer si effettua un controllo del parametro userId. Nel caso in cui fosse nullo, la visita viene creata ma non avrà alcuna relazione con un nodo Customer. Questo vuol dire che le visite non legate ad alcun Customer rappresentano visite in cui l'utente non ha eseguito l'autenticazione.

Successivamente, viene eseguito un ciclo sulla lista delle azioni della visita e per ognuna di esse viene creato un Nodo Action. Utilizzando dei blocchi condizionali in Cypher viene simulato un costrutto if-then-else. Una pagina web viene creata nel caso in cui l'azione sia di tipo "action", ovvero corrisponda alla visualizzazione. Se l'azione è di tipo "ecommerceOrder" si genera di conseguenza la relazione con il nodo Order. Nel caso di azioni di tipo cartUpdate, Download, Outlink o Search si

arricchiesce il nodo Action con delle proprietà aggiuntive.

Infine, eseguendo la funzione collect in una visita, tutte le azioni vengono raggruppate e con dei cicli for annidati viene creata la relazione :NEXT per indicare l'ordine in cui esse avvengono.

```
UNWIND $rows as row

    // website
    MERGE (w:Website {{websiteId: row.{'idSite'}}})
    ON CREATE SET w.name = row.{'siteName'}

    // visit
    CREATE (v:Visit {{visitId: row.{'idVisit'},
        duration: toInteger(row.{'visitDuration'}),
        serverDateTime: datetime({{epochSeconds: toInteger(row.{'firstActionTimestamp'})}}),
        localHour: toInteger(row.{'visitLocalHour'}),
        converted: False,
        referrerType: row.{'referrerType'}
    }})
    CREATE (w)-[:HAS]->(v)

    // customer (if userId is not null)
    FOREACH(i in CASE WHEN row.{'userId'} IS NOT NULL THEN [1] ELSE [] END |
        MERGE (c:Customer {{customerId:row.{'userId'}}})
        CREATE (c)-[:PAYS_A]->(v)
    )

    // device
    MERGE (device:Device {{deviceType: row.{'deviceType'}}})
    MERGE (brand:Brand {{brandName: row.{'deviceBrand'}}})
    MERGE (resolution:Resolution {{value: row.{'resolution'}}})
    MERGE (browser:Browser {{browserId: row.{'browserCode'}+row.{'browserVersion'}}})
    ON CREATE SET browser.name = row.{'browserName'}, browser.version = row.{'browserVersion'}
    MERGE (os:OperatingSystem {{operatingSystemId: row.{'operatingSystemCode'}+row.{'
     operatingSystemVersion'}}})
    ON CREATE SET os.name = row.{'operatingSystemName'}, os.version = row.{'operatingSystemVersion
     '}
    MERGE (device)-[:OF_BRAND]->(brand)
    MERGE (device)-[:WITH_RESOLUTION]->(resolution)
    MERGE (device)-[:WITH_BROWSER]->(browser)
    MERGE (device)-[:WITH_OS]->(os)
    CREATE (v)-[:FROM_DEVICE]->(device)

    // location
    MERGE (loc:Location {{city: row.{'city'}}})
    ON CREATE SET loc.country = row.{'country'},
                  loc.continent = row.{'continent'}
    MERGE (v)-[:FROM]->(loc)

    WITH v, row
    UNWIND row.{'actionDetails'} as action

    CREATE (a:Action {{
                actionId: action.{'pageId'},
                type: action.{'type'},
                dateTime: datetime({{epochSeconds: toInteger(action.{'timestamp'})}})
            }})
```

```
51
52
53
54      // if action type == 'view' --> view webpage
55      FOREACH(i in CASE WHEN action.{'type'} = 'view' THEN [1] ELSE [] END |
56          SET a.timeSpent = toInteger(action.{'timeSpent'})
57          MERGE (page:Webpage {{webpageId: action.{'pageIdAction'}}})
58          ON CREATE
59              SET page.url = action.{'url'},
60                  page.type = action.{'pageType'}
61          CREATE (a)-[:TO_PAGE]->(page)
62      )
63
64      // if action type == 'cartUpdate'
65      FOREACH(i in CASE WHEN action.{'type'} = 'cartUpdate' THEN [1] ELSE [] END |
66          SET a.event = action.{'eventAction'},
67              a.quantity = action.{'eventValue'}
68      )
69
70      // if action type == 'ecommerceOrder'
71      FOREACH(i in CASE WHEN action.{'type'} = 'ecommerceOrder' THEN [1] ELSE [] END |
72          SET a.actionId = action.{'orderId'}
73          SET v.converted = True
74          MERGE (order:Order {{orderId: action.{'orderId'}}})
75          ON CREATE
76              SET order.dateTime = datetime({{epochSeconds: toInteger(action.{'timestamp'})}}),
77                  order.revenue = toFloat(action.{'revenueSubTotal'}),
78                  order.items = toInteger(action.{'items'})
79          CREATE (a)-[:TO_ORDER]->(order)
80      )
81
82      // if action type == 'search'
83      FOREACH(i in CASE WHEN action.{'type'} = 'search' THEN [1] ELSE [] END |
84          SET a.keyword = action.{'siteSearchKeyword'}
85      )
86
87      // if action type == 'outlink' || type == 'download'
88      FOREACH(i in CASE WHEN action.{'type'} = 'outlink' OR action.{'type'} = 'download'
89      THEN [1] ELSE [] END |
90          SET a.url = action.{'url'}
91      )
92
93      CREATE (v)-[:HAS_ACTION]->(a)
94
95      WITH v, collect(a) AS nodes
96      // Relationship between nodes of the current visit
97      FOREACH (i in range(0, size(nodes) - 2) |
98          FOREACH (node1 in [nodes[i]] |
99              FOREACH (node2 in [nodes[i+1]] |
100                 CREATE (node1)-[:NEXT]->(node2))))
101
102     RETURN count(v) as total
```

4.5 Valutazioni e dimensione dei blocchi

Lo schema del database ottenuto è rappresentato in figura 4.1. Il nodo principale è quello delle visite, a cui sono collegate le informazioni riguardo il cliente che l'ha effettuata, il sito su cui è avvenuta, il device utilizzato e la posizione geografica. Ad ogni visita è associata una o più azioni. I nodi Page e Order sono legati alle azioni, rispettivamente di tipo "view" e "ecommerceOrder". Infine, le azioni hanno una relazione tra loro di tipo NEXT, utile per poter rappresentare l'ordine in cui sono avvenute.

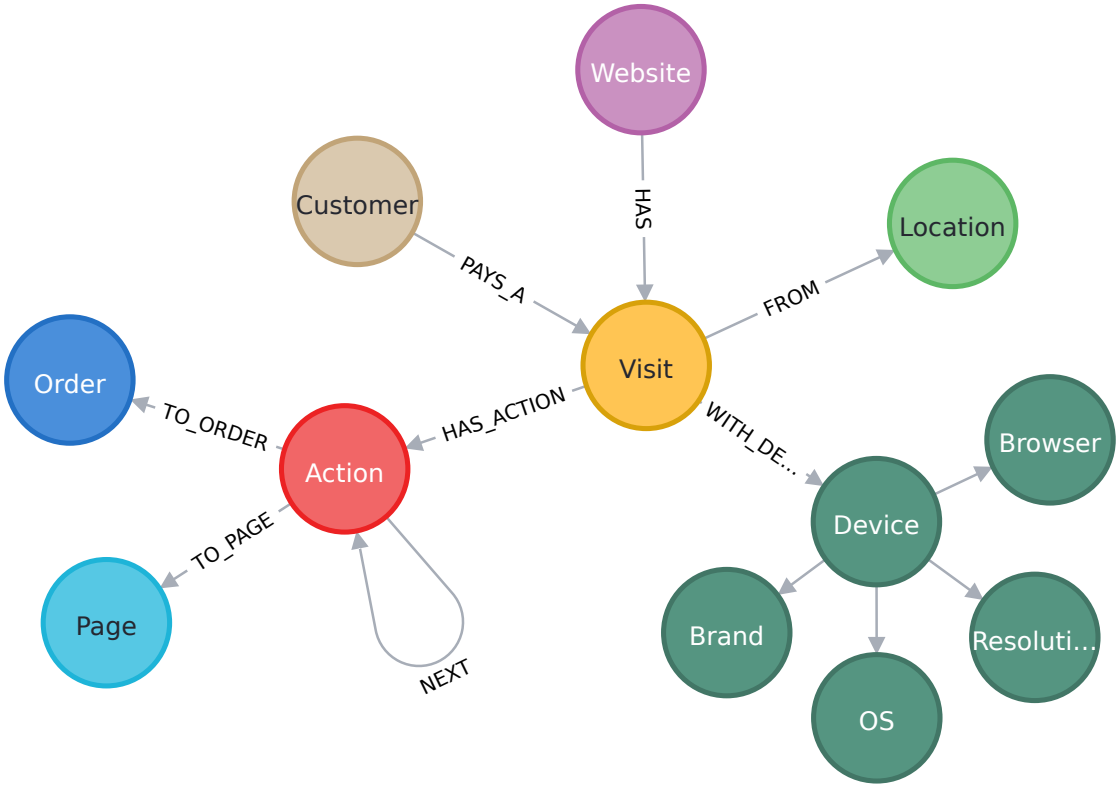

Figura 4.1: Schema del database a grafo

La progettazione della base dati è una fase fondamentale in quanto la definizione dei vincoli d'integrità, con l'implicita creazione degli indici, sulle chiavi primarie permettono di incrementare le prestazioni di salvataggio in modo significativo e consentono di mantenere una base dati consistente a seguito di qualsiasi inserimento o modifica.

Per poter valutare le prestazioni sono stati considerati i dati relativi alle visite nell'ultima settimana di Giugno 2021 (23/06/2021 - 30/06/2021), che ammontano a 30.184 e a cui sono associate 213.116 azioni. Ogni visita è caratterizzata da 103 informazioni, mentre le azioni ne contengono al massimo 36. Nel complesso, questi dati possono essere memorizzati in un file, in formato json, di dimensione 224.5 MB. Il salvataggio dei dati in un DataFrame comporta un uso di memoria di 23.9+ MB per le visite e 58.5+ MB per le azioni.

Dopo aver processato i dati seguendo le fasi descritte nei capitoli precedenti il numero di visite totali è di 18.905, che equivale a una riduzione del 37.4% circa e un numero di colonne pari a 22; mentre il numero di azioni è 200.440 (-6%) con un massimo di 14 colonne. Di conseguenza, l'occupazione in memoria dei DataFrame è diminuita a 3.2+ MB e 21.4+ MB, rispettivamente per il DataFrame delle visite e quello delle azioni. Questi dati essere memorizzati in un file, in formato json, di dimensione pari a 59.9 MB, circa un quarto della dimensione del DataFrame di partenza.

Nelle tabelle 4.5 e 4.6 sono riassunte le informazioni sulle dimensioni dei DataFrame delle visite e delle azioni prima e dopo il processo di trasformazione.

	# Righe	**# Colonne**	**Memoria [MB]**
DataFrame iniziale	30.184	103	23.9+
DataFrame finale	18.905	22	21.4+

Tabella 4.5: Dimensione del DataFrame delle visite prima e dopo il processo di trasformazione

	# Righe	**# Colonne**	**Memoria [MB]**
DataFrame iniziale	213.116	36	58.5+
DataFrame finale	200.440	14	55.4+

Tabella 4.6: Dimensione del DataFrame delle azioni prima e dopo il processo di trasformazione

Considerando le prime 10.000 visite del DataFrame considerato precedentemente (visite nel periodo 23/06/2021 - 30/06/2021) si valuta la prestazione, in termini di tempo impiegato, per eseguire l'inserimento nel database seguendo diversi approcci. Questo sottoinsieme del DataFrame comporta la creazione di 126.514 nodi e 247.057 relazioni. Siccome il numero di azioni non è costante per ogni visita, è più corretto

esprimere le prestazioni in base al numero di nodi e relazioni piuttosto che alle visite create.

L'algoritmo descritto per l'inserimento nella base dati è stato eseguito con una CPU Intel i7 @ 2.20GHz con 16 GB di memoria. Il database è in locale, per questo motivo non è necessario prendere in considerazioni possibili variazioni delle prestazioni a causa della rete, che possono dipendere solo dal sovraccarico della CPU. Per ovviare a questo problema si sono misurati i tempi di esecuzione più volte, con la CPU in condizioni di carico simili, e quindi si sono considerati i valori medi. Inoltre, il database è stato resettato prima di ogni misurazione.

La scelta di operare a blocchi per eseguire il salvataggio si è rivelata corretta: dalla figura 4.2 si osserva che all'aumentare della dimensione dei blocchi si ottengono dei tempi di inserimento migliori. In particolare si nota un miglioramento marcato passando da inserimenti riga per riga (dimensione del blocco pari a uno) e inserimenti con blocchi di dimensione pari a dieci.

Inoltre, la definizione degli indici, con l'implicita creazione dei vincoli di integrità, sulle chiavi primarie ha portato ad un abbattimento dei tempi di inserimento. Le prestazioni migliori si sono ottenute utilizzando blocchi di dimensione pari a 1.000 (in aggiunta all'uso degli indici) con un tempo di inserimento di 15,18 secondi.

Per dimensioni troppo elevate dei blocchi si sono notati peggioramenti nei tempi di inserimenti, raggiungendo tempi anche maggiori degli inserimenti senza blocchi, senza indici.

Considerando il salvataggio a blocchi di dimensione pari a 1.000 si può analizzare l'andamento dei tempi di inserimento durante l'intero processo, suddiviso in dieci iterazioni (salvataggio di 10.000 visite).

In figura 4.3 si nota che senza l'utilizzo degli indici sulle chiavi primarie si ottiene una crescita esponenziale durante l'inserimento delle visite.

Il grafico in figura 4.4 evidenzia una crescita lineare nella variazione dei tempi di inserimento per ciascun blocco: all'aumentare dei dati memorizzati nel database aumenta il tempo di inserimento del blocco.

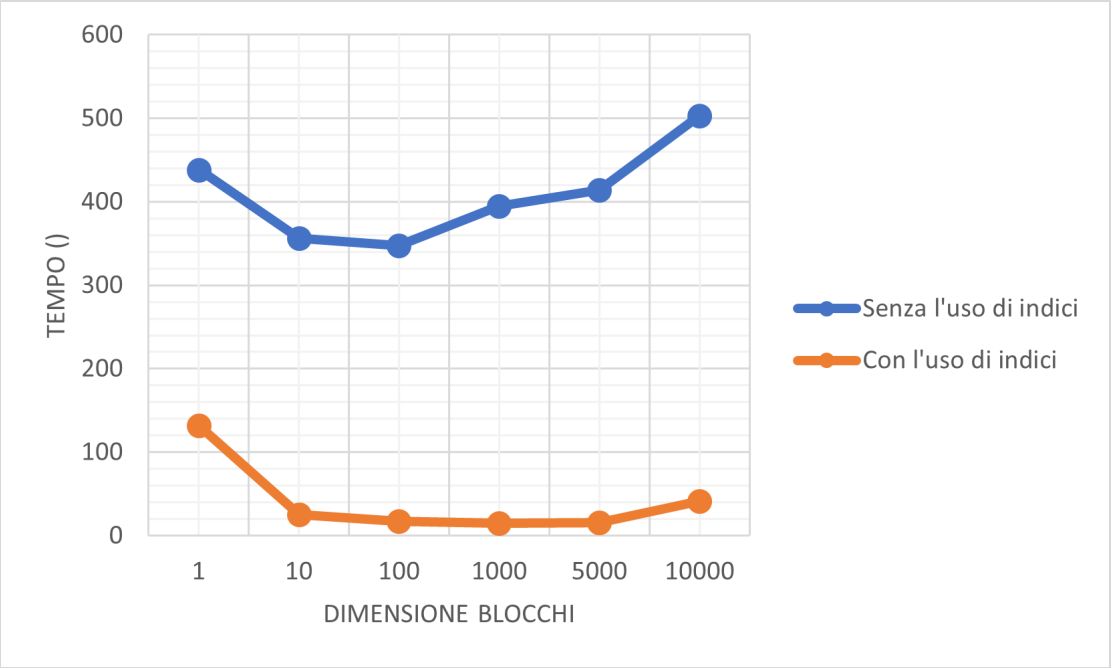

Figura 4.2: Tempi di inserimento a blocchi

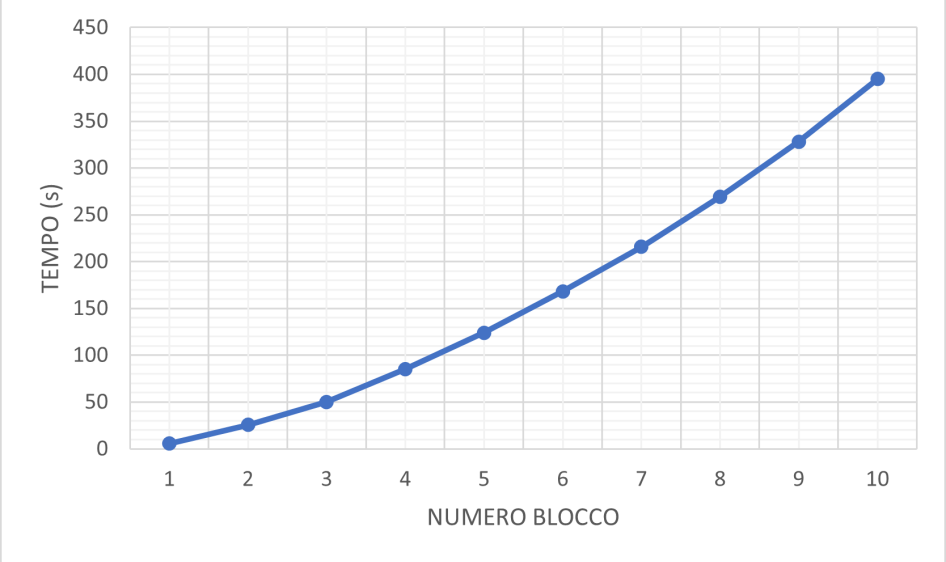

Figura 4.3: Inserimento senza indici con blocchi di dimensione 1000

Invece, utilizzando gli indici sulle chiavi primarie, oltre a un tempo complessivo di inserimento minore, già osservato precedentemente, si osserva una crescita lineare dei tempi di salvataggio (si veda figura 4.5).

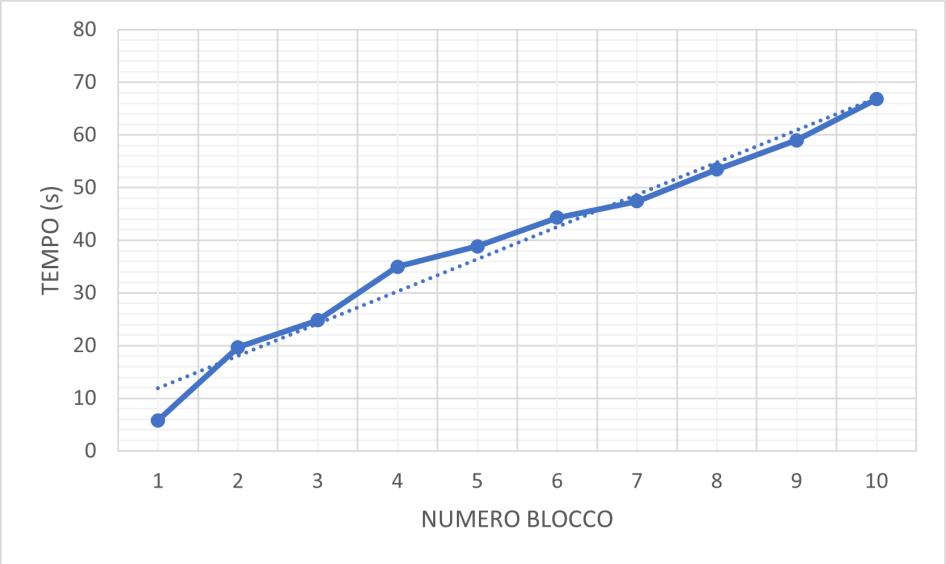

Figura 4.4: Variazione dei tempi di inserimento di ciascun blocco (con blocchi di dimensione 1000)

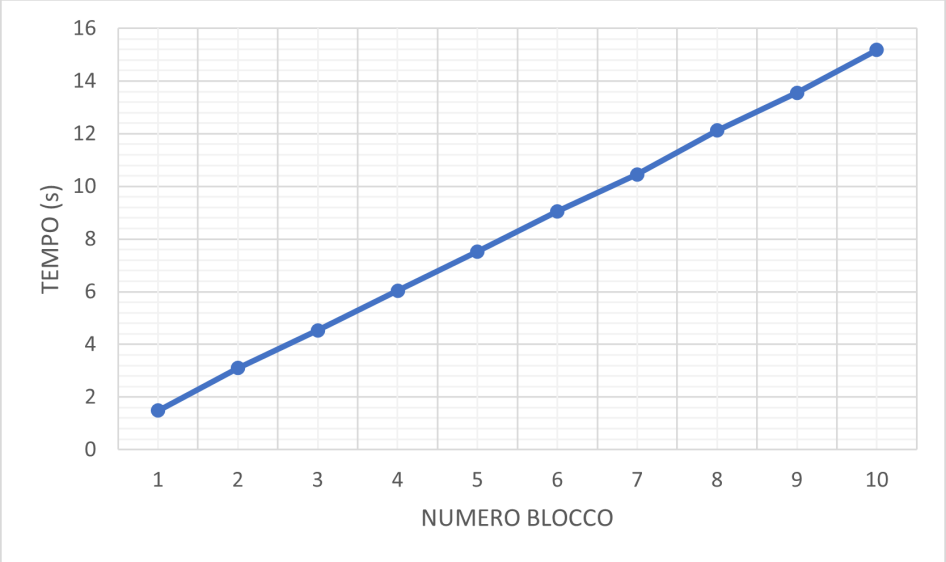

Figura 4.5: Inserimento con indici con blocchi di dimensione 1000

La figura 4.6 mostra la variazione dei tempi di inserimenti per ogni step e ancora una volta si evince che il tempo è circa costante e si attesta a circa 1.5 secondi per ogni blocco (di dimensione 1.000). Si osservi che le piccole variazioni che si sono verificate potrebbero essere causate da una mole di dati diversa da processare in

un blocco rispetto a un altro (perché ciascuna visita contiene un numero variabile di azioni).

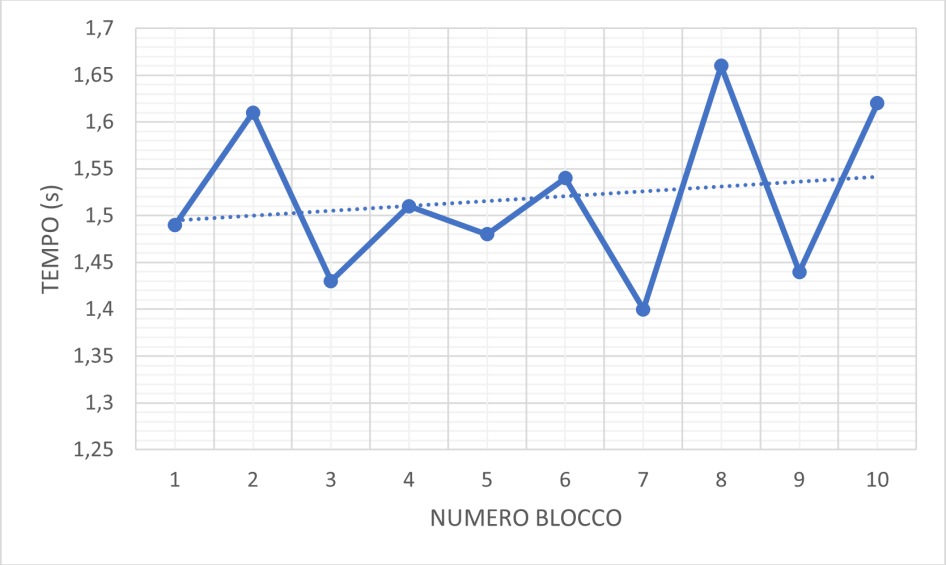

Figura 4.6: Variazione dei tempi di inserimento con indici (con blocchi di dimensione 1000)

Capitolo 5

Implementazione del modello predittivo

Questo capitolo si focalizza sul tema principale della tesi, ovvero quello di riuscire a addestrare un modello di classificazione per poter predire l'intento finale di acquisto di un cliente. Il capitolo prevede una fase iniziale di descrizione del dataset considerato. Prima di procedere all'applicazione degli algoritmi di machine learning è prevista una fase di pre-processing e successivamente una fase in cui si selezionano le features, ovvero le variabili indipendenti del modello che si vuole generare. Nelle ultime sezioni del capitolo si riassumono e valutano i risultati ottenuti.

5.1 Descrizione del dataset

Con riferimento ai dati prodotti dalla navigazione dei clienti, nel periodo che va dal 23 giugno 2021 al 23 agosto 2021, su un e-commerce B2C che si occupa della vendita di prodotti di bellezza e per la cura del corpo, il software di tracking Matomo ha collezionato un totale di 152.424 visite, di cui 3.291 corrispondono a sessioni d'acquisto. Tutte queste visite sono state memorizzate nel database a grafo utilizzando la metodologia descritta nel capitolo 4.

Con delle semplici interrogazioni alla base dati, si osserva che il numero di visite in cui l'utente ha eseguito il log-in è 76.290, e di conseguenza, le rimanenti 76.134 visite sono di utenti non autenticati.
Ciò che si osserva fin da subito è che le sessioni non autenticate non generano valore, in quanto è necessario che un cliente si autentichi affinché possa visualizzare i dettagli dei prodotti o eseguire qualsiasi azioni. Per questo motivo il numero di visite autenticate a cui è associato un ordine è uguale al numero di visite convertite (3.291), mentre il numero di visite non autenticate coincide con il numero di visite

non autenticate e non convertite (76.134); infine, il numero di conversioni nelle visite non autenticate è pari a zero. La tabella 5.1 riassume il numero di visite convertite e non convertite in relazione al fatto che un cliente si sia autenticato o meno.

	Convertite	**Non convertite**
Autenticate	3.291	72.999
Non autenticate	0	76.134

Tabella 5.1: Distribuzione del numero di visite

5.2 Pre-processing

Come osservato nelle sezioni iniziali di questo capitolo, le sessioni non autenticate non sono utili al fine dell'obiettivo prefissato. Per questo motivo tutte queste visite, con le relative azioni, sono state eliminate. In seguito a questa operazione preliminare il numero di visite convertite rimane invariato e si ha quasi un dimezzamento nella dimensione delle visite non convertite, e quindi in generale di tutto il dataset.

Tra tutte le visite sono state rimosse anche tutte quelle che hanno informazioni su subscriptions perché questi non sono veri e propri ordini materiali ma abbonamenti per mostrare il catalogo dei prodotto (B2B). Per fare questo si sono eliminate le visite con azioni legate e pagine contenti nell'URL la parola subscription. In questo modo si rimuovono 1.554 visite non convertite e 173 convertite. Tali conversioni non sono utili al modello predittivo.

Le sessioni effettuate da amministratori del sito web sono caratterizzate dalla visita alle pagine che contengono nell'URL la stringa "adminui", per ovvie ragioni anche queste visite sono state rimosse dal dataset, non influendo sul numero di visite convertite. Il numero di visite si riduce di 147.

Si sono osservati i path più seguiti per arrivare ad effettuare un ordine sul sito e-commerce. La tabella 5.2 mostra un'informazione interessante: l'azione precedente a un ordine è nel 92% dei casi l'azione di visita alla pagina di ordine completato ("view-checkout_done"). Questo potrebbe essere un possibile errore derivato dal caricamento dei dati nel database. Per questo motivo, grazie alla semplicità d'uso del database non relazionale, sono stati analizzati nel dettaglio i percorsi più frequenti per arrivare all'ordine, considerando più azioni e non solo

:Action.type	COUNT(:Action.type)
view-checkout_done	5.136
view-checkout_payment	353
view-user_profile	39
view-cart	16

Tabella 5.2: Azioni precedenti all'ordine, ordinate per frequenze descrescenti

quella precedente. Ciò che si nota è che l'azione di ordine e-commerce non si trova sempre in posizione corretta e questo errore è causato dal fatto che il timestamp di questa azione è uguale a quello dell'azione di visita di completamento dell'ordine e durante il salvataggio si sono considerati altri parametri per ordinare queste azioni. Ciascun ordine deve essere preceduto dall'azione di visita alla pagina di completamento dell'ordine.
I percorsi d'acquisto più frequenti, dopo aver effettuato un riordino delle azioni per correggere l'errore descritto precedentemente, sono riassunti nella figura 5.1.

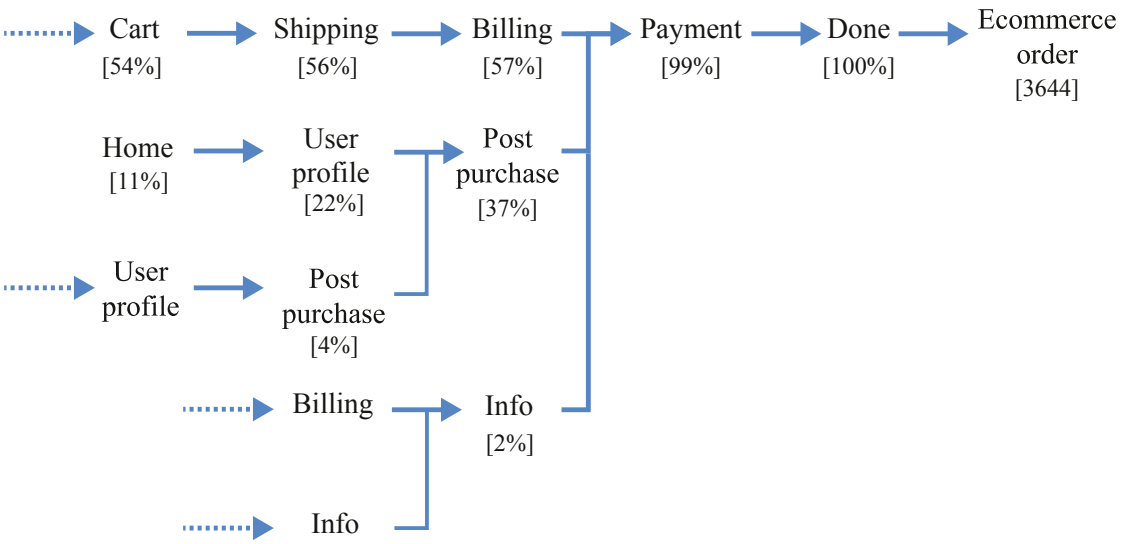

Figura 5.1: Percorsi d'acquisto più frequenti

In modo analogo si osserva anche che ci sono sequenze di azioni riconducibili a un ordine, senza però l'azione conclusiva "ecommerce_order" . Queste situazioni sono causate da un malfunzionamento del software di tracking. In particolare è stato aggiunto l'evento di ordine in seguito a sequenze di azioni nel seguente ordine:

view-checkout_payment –> view-checkout_done. (vscode punto 4). Il numero di visite convertite aumenta a 3.644.

Il numero di ordini è maggiore del numero di visite convertite, in quanto sono presenti visite con più ordini. In questo caso sono state rimosse tutte le azioni successive al primo ordine. Il numero di visite, in seguito a queste modifiche, rimane invariato.

5.2.1 Analisi della lunghezza delle visite

L'obiettivo principale è quello di riuscire a classificare una visita dopo un certo numero di interazioni del cliente con il sito web. Per poter scegliere in modo opportuno tale numero si è deciso di analizzare la distribuzione della lunghezza delle visite, in termini di azioni eseguite. La tabella 5.3 riassume le informazioni principali sulla distribuzione dei valori del numero di azioni eseguite in ciascuna visita e anche nello specifico nelle visite convertite o non convertite.

	Tutte le visite	**Visite convertite**	**Visite non convertite**
mean	14,63	21,50	14,28
std	25,00	34,39	24,36
min	2	3	2
25%	4	6	4
50%	7	11	7
75%	15	20	15
max	500	423	500

Tabella 5.3: Distribuzione della lunghezza delle visite

Come è già stato osservato precedentemente il dataset presenta una distribuzione dei dati non bilanciata, con un numero elevato di sessioni non convertite e perciò nella tabella si nota subito come i valori nella colonna delle azioni delle visite non convertite sono molto simili ai valori della colonna che raggruppa tutte le visite.
Le visite che sono terminate con un acquisto presentano un numero di azioni medio pari a 21,50 e una deviazione standard di 34,39. I valori massimi si discostano significativamente dal valore medio e perciò potrebbero indicare dei valori anomali. Infine, un'informazione importante si ottiene dai valori del primo, secondo e terzo quartile: il 25% delle visite convertite hanno un numero di azioni pari a 6, il 50% si assesta a 11 azioni e il 75% a 20 azioni; rispetto alle visite non convertite, tali valori sono leggermente maggiori, ovvero molte delle sessioni non convertite presentano

un numero di azioni minori.

Il grafico a scatola e baffi e l'istogramma sono un ottimo strumento per visualizzare le informazioni riassunte nella tabella precedente. a capire la distribuzione. Il primo grafico (5.2) mette a confronto la distribuzione delle lunghezze delle visite per i tre diversi dataset, mentre l'istogramma evidenzia maggiormente quanto detto precedentemente: le visite convertite sono caratterizzate da un numero maggiore di azioni; in particolare il 25% delle visite non convertite hanno un numero di azioni massimo pari a 4.

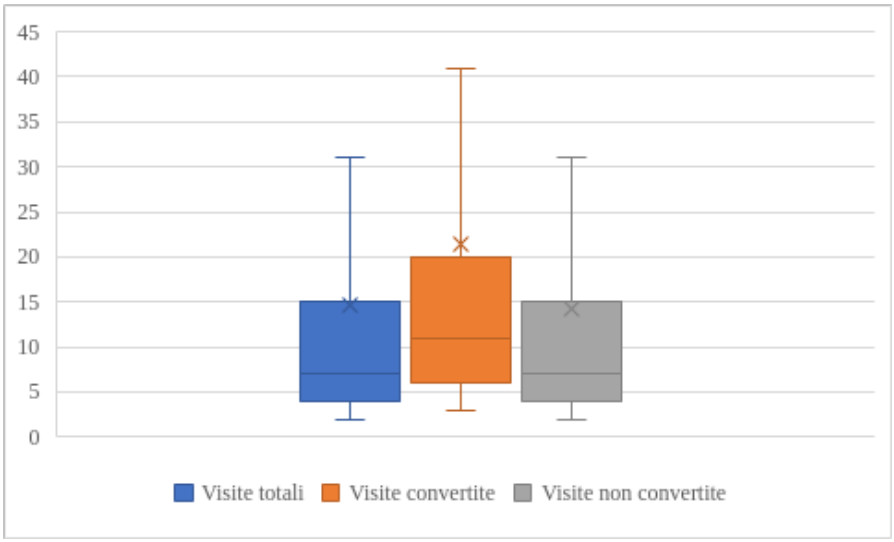

Figura 5.2: Boxplot della lunghezza delle visite non convertite

In seguito a queste osservazioni si è deciso che il numero di azioni dopo cui l'algoritmo classifica una visita è dieci, un numero sufficientemente grande per avere alcune informazioni delle azioni eseguite inizialmente ed è un numero non troppo grande per poter mostrare del contenuto personalizzato ai clienti in base al risultato della classificazione.

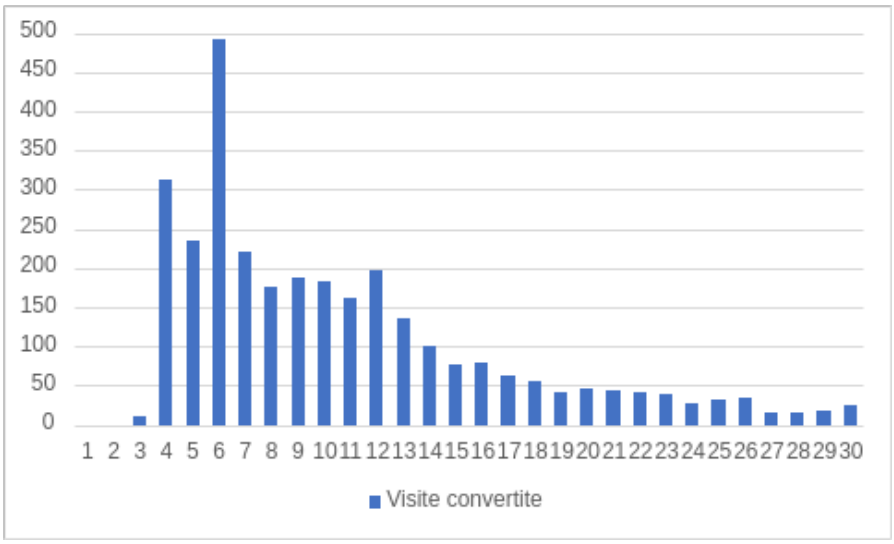

Figura 5.3: Istogramma della lunghezza delle visite convertite

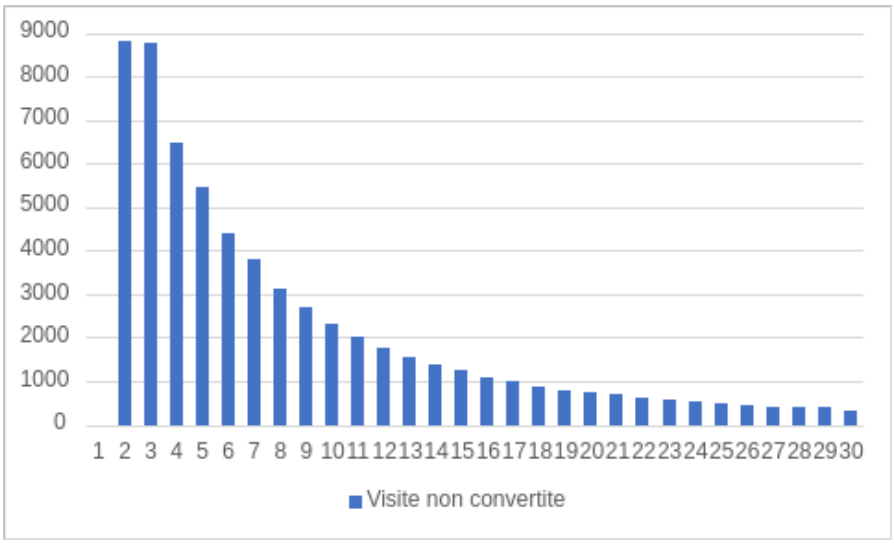

Figura 5.4: Istogramma della lunghezza delle visite non convertite

In conclusione, considerando solo le visite che hanno almeno un numero di azioni pari a dieci si riduce leggermente il livello di sbilanciamento del dataset: il numero di visite convertite è di 2.010, mentre il numero di visite non convertite è di 27.147.

5.3 Analisi del dataset

In questa sezione si effettua un'analisi delle informazioni presenti nel dataset, esplorando le distribuzioni delle variabili indipendenti (esogena) e quindi osservando il loro impatto sulla variabile dipendente (endogena). Questa fase permette di comparare le sessioni d'acquisto con quelle non d'acquisto in relazione a dei parametri come la lunghezza delle visite, il device utilizzato ecc.

Per prima cosa si vuole analizzare l'impatto delle caratteristiche temporali delle visite, ovvero il giorno della settimana e l'ora del giorno in cui sono avvenute.
La figura 5.5 mostra il rapporto tra il numero di visite avvenute in un determinato giorno rispetto al numero di visite totali, separatemene per le visite convertite e quelle non convertite. Nel grafico si osserva che circa il 20% delle visite convertite si verifica nella prima giornata della settimana, mentre si ha un picco minimo il sabato. Le visite non convertite si distribuiscono nei giorni settimanali in modo simile a quella convertite, ciò permette di capire che le giornate del week end sono quelle con meno traffico nel sito.

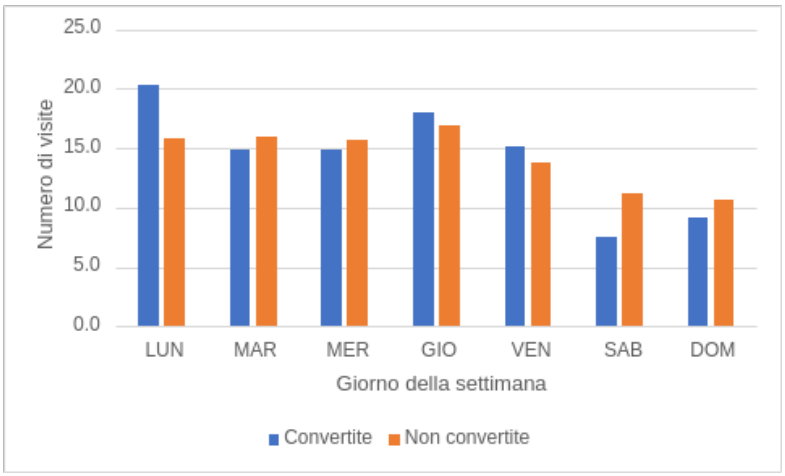

Figura 5.5: Rapporto tra le visite convertite e non convertite rispetto al numero totale di visite nei giorni della settimana

La distribuzione delle visite nelle ore del giorno sono mostrate in figura 5.6. Questa informazione si riferisce all'ora di inizio della visita ed è rappresentata seguendo il fuso orario dell'Europa centrale (CET). Come da aspettative, durante le ore notturne si verifica il minor numero di visite. Questo perché la maggior parte delle visite in questo e-commerce B2B avvengono in un paese europeo. Durante le ore del mattino si verifica il maggior numero di ordini, con un picco alle ore

9:00. Le visite non convertite si distribuiscono in modo omogeneo durante l'intera giornata.

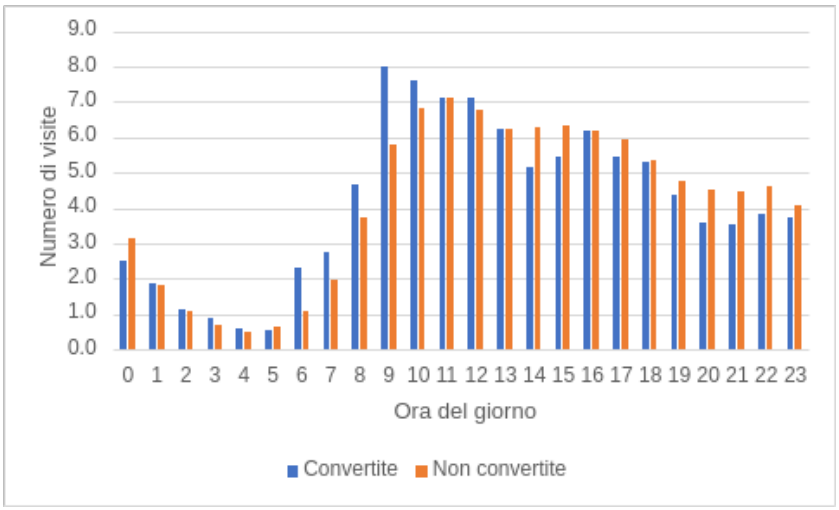

Figura 5.6: Rapporto tra le visite convertite e non convertite rispetto al numero totale di visite nelle ore del giorno

Dalla figura 5.6 si osserva che si ha una netta predominanza del canale d'ingresso di tipo "direct" tra le visite convertite: circa il 65% delle visite che si sono concluse con un ordine sono caratterizzate da questo canale di ingresso. Il canale diretto è il più utilizzato anche nelle visite non convertite, seguito dal tipo "campaign" , non molto frequente nelle visite convertite.

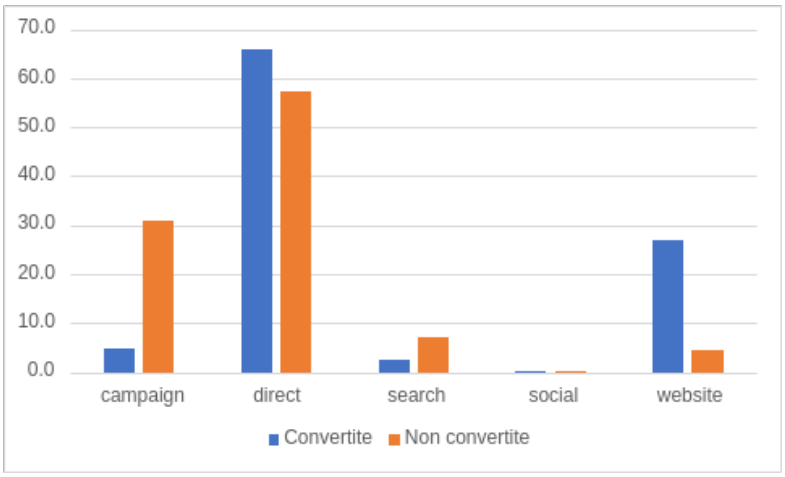

Figura 5.7: Rapporto tra le visite convertite e non convertite rispetto al numero totale di visite per ogni tipo di canale di ingresso

La figura 5.8 mostra come il dispositivo maggiormente usato nelle visite è di tipo “desktop” e di tipo “smartphone” (14%). mostrano come i dispositivi desktop e smartphone siano quelli maggiormente usati per accedere al sito web. Mentre per le visite non convertite si ha un equilibrio nella frequenza d’uso dei due dispositivi, per le sessioni convertite si nota un netto distacco tra i due tipo: il numero di visite convertite con l’uso di un dispositivo “desktop” è di circa 85%. Ciò significa che gli utenti navigano sul sito con diversi dispositivi, ma preferiscono comprare su dispositivi fissi.

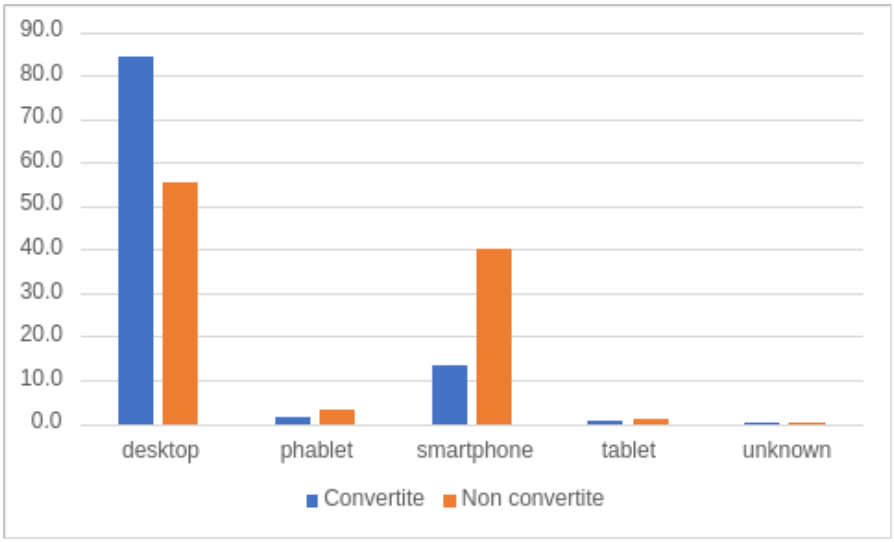

Figura 5.8: Rapporto tra le visite convertite e non convertite rispetto al numero totale di visite per ogni tipo di device utilizzato

Infine, confrontando la figura 5.8 con la figura 5.9 si può osservare come l’andamento dei due grafici è molto probabile che i dispositivi “Desktop” siano associati con il sistema operativo “Windows” e i dispositivi “Smartphone” con il sistema operativo “Android” . L’unica informazione che offre questo secondo grafico è che i clienti del dataset preferiscono utilizzare dispositivi Windows e Android piuttosto che dispositivi iOS, tuttavia questa informazione potrebbe essere fuorviante, in quanto essa potrebbe essere legata al dataset analizzato (nel periodo indicato) e non a ciò che accade realmente.

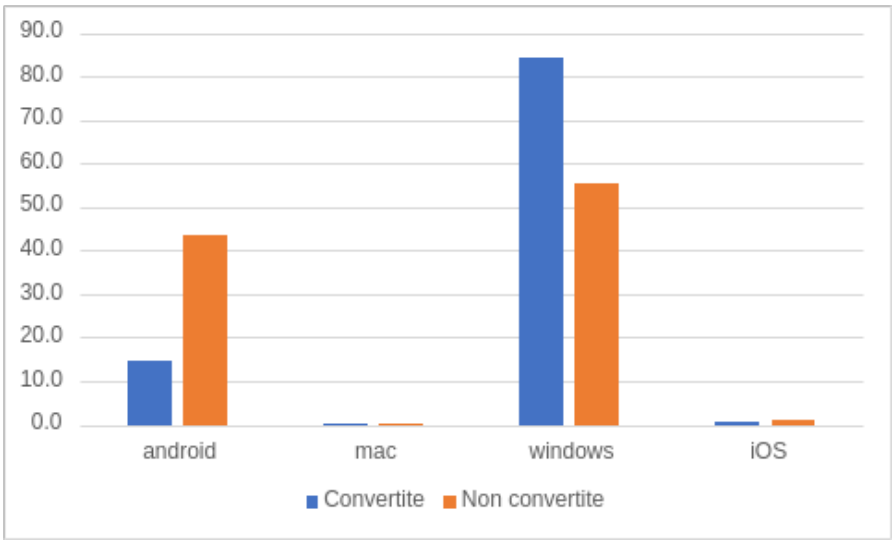

Figura 5.9: Rapporto tra le visite convertite e non convertite rispetto al numero totale di visite per ogni sistema operativo utilizzato

5.4 Progettazione e selezione di feature

In questa sezione si descrivono due processi molto importanti nel machine learning: il processo di feature engineering e quello di feature selection. Al termine dei due processi si ottiene un set di funzionalità che saranno utilizzate dagli algoritmi di machine learning.

5.4.1 Feature Engineering

Il processo di feature engineering, o di progettazione di nuove funzionalità, sfrutta le conoscenze del dominio per poter aumentare il numero di funzionalità al fine di ottenere un modello predittivo più accurato e performante. Tali funzionalità hanno l'obiettivo principale di aggiungere informazioni non presenti e non correlate con le funzionalità del dataset originale. Con una buona progettazione di nuove funzionalità si ottiene un modello più complesso ma anche più facilmente interpretabile [21].

Il numero totale di funzionalità considerate per questo studio è 14 e sono riassunte nella tabella 5.4, in cui è specificato il nome, il tipo di dato e una breve descrizione.

Nome	**Valori**	**Descrizione**
converted	True/False	Visita convertita? È la variabile dipendente del modello predittivo
dayOfWeek	1-7	Giorno della settimana in cui è avvenuta la visita
hourOfDay	0-23	Ora del giorno in cui è avvenuta la visita
referrerType	string (campaign, direct...)	Canale di ingresso utilizzato dal cliente
deviceType	string (desktop, smartphone...)	Device utilizzato dal cliente durante la visita
visitorType	string (new, returning)	Tipo di cliente, può assumere solo due valori
os	string (android, iOS...)	Sistema opertivo utilizzato durante la visita
purchaseIndex	float (0-1)	Rapporto tra il numero di ordini e il numero di visite
daysSinceLastOrder	int	Giorni dall'ultimo ordine
daysSinceLastVisit	int	Giorni dall'ultima visita
itemsInCart_10	int	Numero di articoli nel carrello dopo 10 azioni
duration_10	int	Durata (secondi) della sessione dopo 10 azioni
searches_10	int	Numero di ricerche dopo 10 azioni
products_10	int	Numero di prodotti visit dopo 10 azioni

Tabella 5.4: Lista delle feature selezionate

Alcune di queste funzionalità sono state aggiunte specificamente per classificare le visite dopo che un cliente ha eseguito dieci azioni, e si distinguono dalle altre in quanto hanno come prefisso la string "_10" .

L'aggiunta delle nuove funzionalità è stata effettuata direttamente in neo4j sfruttando la semplicità d'uso della base dati a grafo nel gestire le relazioni. Per esempio la feature riguardanti il numero di azioni è stata ricavata utilizzando la relazione "HAS_ACTION" presente tra i nodi di tipo "action" e quelli di tipo "visit".

5.4.2 Feature Selection

Il processo di selezione delle funzionalità ha il compito principale di limitare il numero di funzionalità, perché gli algoritmi di machine learning non funzionano bene quando il numero di funzionalità è elevato. Per questo motivo, tale processo risulta utile quando si deve gestire un numero elevato di feature, specialmente se il numero di caratteristiche supera il numero dei dati [21].

Con l'utilizzo della libreria XGBoost è stata valutata l'importanza delle feature descritte precedentemente (tabella 5.4). La metrica utilizzata per la valutazione è quella del guadagno, in inglese "gain", in quanto essa esprime il contributo relativo della feature per la generazione del modello. Un valore di guadagno alto indica una maggiore importanza della feature. L'immagine 5.10 mostra in ordine decrescente l'importanza delle feature analizzate.

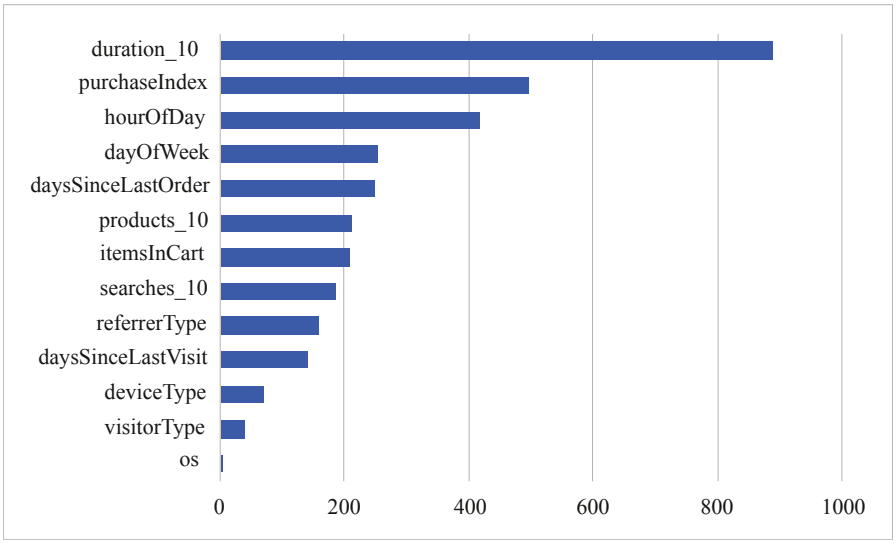

Figura 5.10: Grafico che mostra l'importanza delle feature prodotto da XGBoost

Si osserva che la feature più importante è quella relativa alla durata della sessione dopo 10 azioni. In seguito, si trova il purchaseIndex, che si basa sulle interazioni passate del cliente (rapporto tra il numero di acquisti e visite), e l'ora del giorno in

cui avviene la visita.

In conclusione, si è deciso di mantenere tutte le feature ad esclusione del sistema operativo, che ha un valore molto basso di gain, e la feature "hourOfDay", in quanto è una variabile categorica caratterizzata da un numero elevato di categorie (valore intero nell'intervallo [0, 23]) che aumenterebbe notevolmente la complessità.

5.4.3 Variabili categoriche

Gli algoritmi di classificazione richiedono che i parametri in input e output siano di tipo numerico. Perciò, le variabile categoriche in un dataset necessitano di essere codificate in numeri per poter essere utilizzate. Tra le feature selezionate, quelle categoriche sono: dayOfWeek, referrerType, deviceType, visitorType, os.
La libreria scikit-learn offre diverse soluzioni [22]. La prima si basa sull'utilizzo della classe **LabelEncoder** che permette di mappare ciascuna categoria in un valore numerico. Questo approccio funziona bene in molti casi ed è molto semplice, tuttavia i valori numerici sono sequenziali e ciò potrebbe causare problemi se si utilizzo classificatori che si basano su valori reali, in quanto essi considerano i numeri simili in base alla loro distanza. Per questo motivo esiste una seconda soluzione, chiamata codifica one-hot, in cui si utilizza la classe **OneHotEncoder**, in cui per ciascuna categoria viene create una nuova feature, ovvero si crea una variabile fittizia, detta "dummy". Il risultato è un insieme di variabili dummy che contengono tutte il valore zero tranne quella corrispondente alla categoria corretta, che contiene il valore uno. L'aspetto negativo di questa soluzione è l'aumento del numero di feature.

5.4.4 Valori mancanti

I valori mancanti sono uno dei problemi da risolvere per poter eseguire correttamente un algoritmo di Machine Learning. Il problema dei valori mancanti può essere corretto seguendo diverse strategie [22]. La soluzione più drastica è quella di rimuovere completamente il record dal dataset ed è adottata principalmente quando il dataset ha dimensioni elevate ed è difficile predire il valore mancante. Un'altra soluzione è quella di utilizzare una strategia supervisionata per predire il valore mancante attraverso un modello predittivo. L'ultima soluzione consente di sostituire il valore mancante con un valore ben definito; tale valore può essere un valore diverso dai valori presenti nel dataset oppure utilizzare il valore medio, mediano o quello più frequente.

Nel dataset analizzato sono presenti due feature contenenti valori mancanti: "daysSinceLastOrder" e "daysSinceLastVisit". Esse esprimono il numero di giorni dall'ultimo ordine o dall'ultima visita e la soluzione adottata è quella di inserire un valore di default (-1) non presente negli altri record del dataset.

5.5 Algoritmi di Machine Learning per la classificazione

Per poter generare un modello predittivo delle visite sono stati considerati diversi algoritmi di classificazione supervisionati, tra cui gli algoritmi decision tree, random forest e k-NN.
Seppure fossero tutti algoritmi validi, la scelta finale è ricaduta sull'algoritmo di random forest, che mostra tempi di assegnazione della classe più lenti rispetto all'algoritmo di decision tree, ma permette di avere un'accuratezza elevata. Tale algoritmo si presta molto bene per risolvere casi di classificazione, come quello descritto in questo progetto di tesi.

5.5.1 Metodo di valutazione

Il metodo di valutazione dei modelli si basa sul valore dell'accuratezza e della misura F1. Il codice 5.5.1 mostra la funzione utilizzata per effettuare la valutazione di un modello.

```
1  def evaluate(model, test_features, test_labels):
2      predictions = model.predict(test_features)
3      np_test_labels = np.array(test_labels)
4
5      accuracy = metrics.accuracy_score(np_test_labels, predictions)
6      precision = metrics.precision_score(np_test_labels, predictions)
7      recall = metrics.recall_score(np_test_labels, predictions)
8      F1 = 2 * (precision * recall) / (precision + recall)
9      metric_list = [accuracy, precision, recall, F1]
10
11     return metric_list
```

Il punteggio F1 è una delle misure più utilizzate nei modelli di classificazione. Essa è calcolata come la media armonica di precisione e recupero.

5.5.2 Algoritmo di random forest

Per implementare un classificatore di tipo random forest è stata utilizzata la classe RandomForestClassifier della libreria sklearn. Questa tecnica di machine learning si basa sull'aggregazione di diversi alberi decisionali.

Con il processo di hyperparameter tuning è possibile trovare la combinazione di iperparametri ottimali per poter generare un buon modello di classificazione. Un iperparametro è una variabile che condiziona il processo di training ed ha un impatto significativo sul risultato finale. La metodologia utilizzata per poter eseguire questo processo è mostrata in figura 5.11

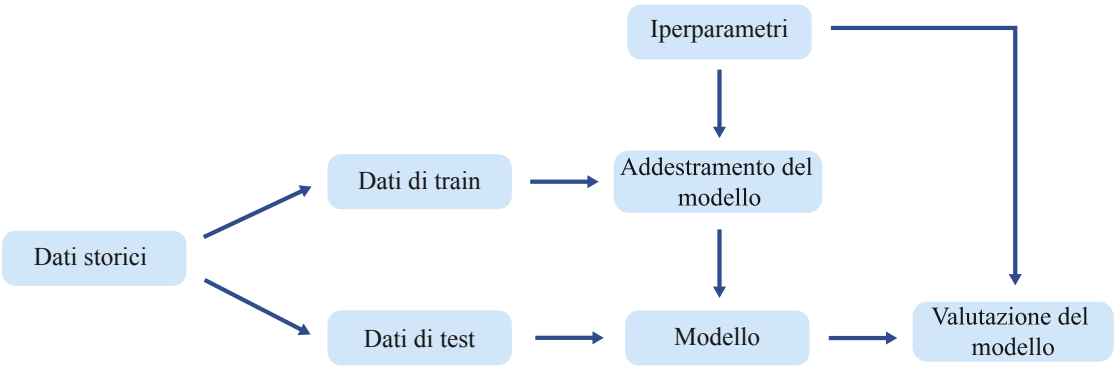

Figura 5.11: Processo di hyperparameter tuning

L'algoritmo random forest ha un set predefinito di iperparametri, osservabili nella documentazione online della libreria sklearn [23]:

- n_estimators = 100
- criterion = 'gini'
- max_depth = None
- min_samples_split = 2
- min_samples_leaf = 1
- min_weight_fraction_leaf = 0.0
- max_features = 'auto'
- max_leaf_nodes = None
- min_impurity_decrease = 0.0
- bootstrap = True
- oob_score = False
- n_jobs = None
- random_state = 42
- verbose = 0
- warm_start = False

- class_weight = None
- ccp_alpha = 0.0
- max_samples = None

Il modello di classificazione di base presenta un'accuratezza del 95.54% e una misura F1 pari a 60.98%.

Si nota fin da subito che il numero di iperparametri è elevato e provare tutte le combinazioni richiederebbe molte risorse e tempo, perciò si è deciso di selezionare solo quelli più importanti:

- n_estimators: numero di alberi decisionali
- max_depth: livello massimo degli alberi decisionali
- min_samples_split: numero minimo di campioni in un nodo per diramare un nodo
- min_samples_leaf: numero minimo di di campioni necessari per poter avere un nodo foglia (ovvero per poter eseguire una divisione del nodo corrente)
- bootstrap: metodologia per la scelta dei campioni; se è "true" i campioni vengono vengono estratti con la politica di rimpiazzo

La classe RandomizedSearchCV permette di eseguire il processo di hyperparameter tuning, selezionando in modo casuale i parametri, da un set pre-definito di possibili valori. Tale classe esegue la validazione incrociata di tipo k-fold per valutare le varie combinazione dei parametri. Nel codice 5.5.2, è stata definita una griglia che genera potenzialmente un numero di combinazioni differenti pari a 1440. La classe RandomizedSearchCV non testa tutte le possibili combinazioni ma solo un suo sottoinsieme, scelto in modo casuale (tale classe accetta come parametro il numero di interazioni, che equivale al numero di combinazioni da considerare). Inoltre, è possibile definire il parametro k della convalidazione incrociata, che definisce il numero di suddivisioni del dataset originale per poter eseguire il test e la valutazione, tenendo presente che un valore elevato di k diminuisce la probabilità di overfitting, ma aumenta il tempo di esecuzione.

```
# Definizione della griglia casuale iniziale
random_grid = {
    'n_estimators': [100, 200, 400, 600, 800, 1000],
    'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
    'min_samples_split': [2, 4, 6],
    'min_samples_leaf': [1, 2, 4, 6],
    'bootstrap': [True, False]
}

# Creazione del modello di classificazione di base
rf = RandomForestRegressor()
# Ricerca casuale dei parametri: 3-fold cross validation con 100
    iterazioni (100 diverse combinazioni)
rf_random = RandomizedSearchCV(
    estimator = rf,
    param_distributions = random_grid,
    n_iter = 100,
    cv = 3,
    random_state=42)
# Fit del modello di ricerca casuale
rf_random.fit(train_features, train_labels)

```

Di seguito è riportata la lista dei migliori parametri generata dall'esecuzione casuale delle varie combinazioni in 300 iterazioni con una validazione di tipo 3-fold.

- n_estimators: 200
- max_depth: 80
- min_samples_split: 6
- min_samples_leaf: 4
- bootstrap: False

Questa prima ricerca ha generato un miglioramento nella misura F1 del 1.23%. L'accuratezza è salita a 95.82%, la misura F1 è pari a 61.73%.

La ricerca casuale aiuta a restringere il campo di ricerca dei migliori parametri: a partire dai parametri ritornati dall'esecuzione precedente, si può definire un nuovo range ridotto e quindi esplorabile interamente. La classe GridSearchCV funziona in modo simile alla classe RandomizedSearchCV, con la sola differenza che essa testa tutte le possibili combinazioni. La nuova lista dei parametri è la seguente:

- n_estimators': [100, 200, 300]
- max_depth: [70, 80, 90]
- min_samples_split: [4, 6, 8]
- min_samples_leaf: [2, 3, 4]
- bootstrap: [True]

Il numero di tutte le combinazioni è di 81, ed è un numero sufficientemente piccolo per poter provare tutte le combinazioni. Eseguendo il metodo fit della classe GridSearchCV con una validazione incrociata di tipo 3-fold, si ricava la miglior combinazione di iperparametri:

- n_estimators = 100
- max_depth = 70
- min_samples_split = 6
- min_samples_leaf = 2
- bootstrap = True

Tale combinazione genera un ulteriore aumento, rispetto al modello di base, dell'accuratezza e nella misura F1 (+1.47%).

5.6 Risultati

La dimensione elevata di ciascun albero del modello random forest, causata dal numero di feature e dalla profondità, non permette di visualizzare e analizzare graficamente i risultati. Per semplificare la visualizzazione si può limitare la profondità degli alberi. In figura 5.12 è rappresentata una porzione di uno degli alberi decisionali. Essa permette di interpretare il modello generato.

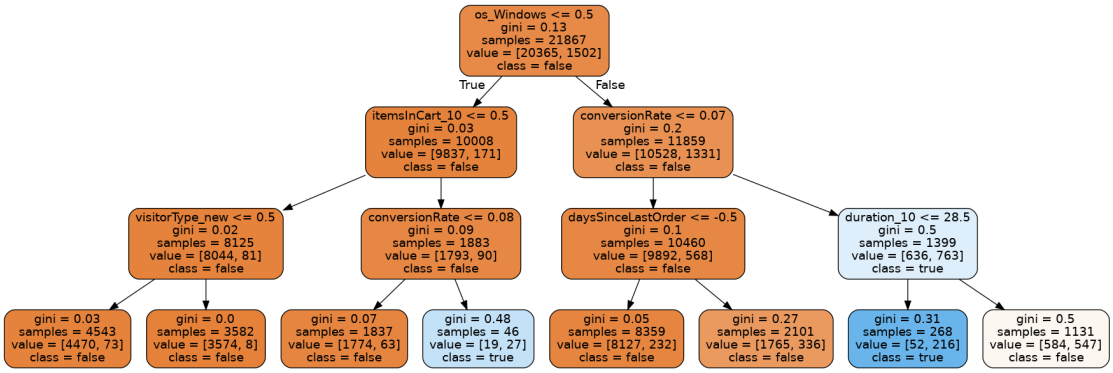

Figura 5.12: Visione parziale di uno degli alberi decisionale del modello random forest

Tutti i nodi, ad esclusione dei nodi foglia, contengono 5 valori:

1. Condizione sulla feature del nodo per poter eseguire uno divisione in altri due nodi.
2. Gini: metrica che descrive la purità di un nodo. Un nodo con un valore di gini uguale a zero indica che il nodo è puro, mentre un nodo con un valore

maggiore di zero implica che all'interno del nodo ci sono campioni che ricadono in classi diverse (il nodo è impuro).
3. Samples: numero di campioni nel nodo. Scendendo di livello il numero di campioni diminuisce.
4. Value: lista contenente la quantità di campioni per ciascuna classe, rispettivamente la classe false (visita senza acquisti) e true (visita con acquisti). La somma dei valori della lisa coincide con la quantità di campioni nel nodo.
5. Class: valore che indica la predizione di classe in quel nodo. Tale classificazione viene ricavata dalla lista descritta precedentemente: la classe che è presente maggiormente è scelta come classe del nodo.

In conclusione, l'accuratezza del modello generato è del 95.78% e la misura del punteggio F1 è del 61.88%. La presenza dominante di visite senza acquisti nel dataset analizzato influisce sul risultato ottenuto. Infatti, dalla figura 5.13, in cui è rappresenta la matrice di confusione, si osserva che la maggior parte delle predizioni sono fatte alla classe false, ovvero quella delle visite senza acquisti. La misura F1 presenta un valore minore in quanto combina i valori di precisione e richiamo di entrambe le classi e quindi mette in risalto il fatto che si stia analizzando un dataset sbilanciato.

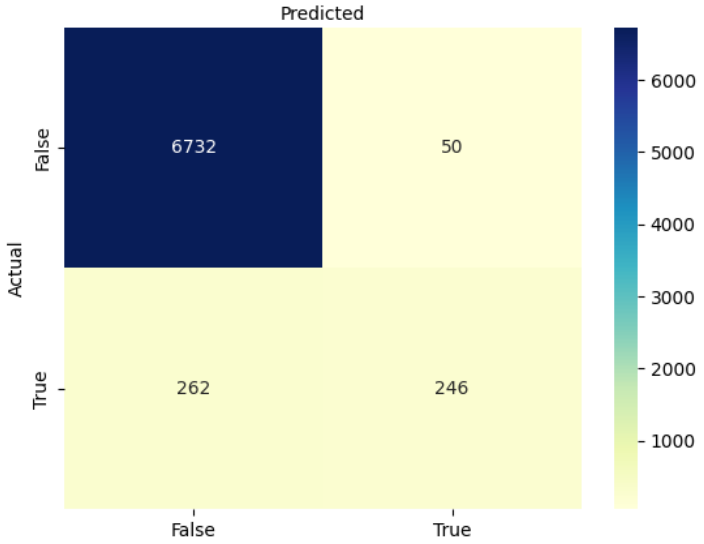

Figura 5.13: Matrice di confusione (random forest)

5.7 Valutazioni

In questa sezione si valutano i risultati mostrati nella sezione precedente, analizzano nel dettaglio i dati mostrati nella matrice di confusione (figura 5.13), facendo riferimento anche a situazioni reali, in cui si potrebbe personalizzare la navigazione dei clienti sul sito e-commerce in base al valore della classificazione.

Seppur il modello predittivo riesca a distinguere discretamente bene i casi di sessioni di compratori dalle sessioni di visitatori, non si ha una netta differenza tra i casi "true positive" e i casi "false negative", ovvero le sessioni dei compratori sono predette correttamente e scorrettamente in una quantità circa uguale. Questo problema è causato dalla natura del dataset, in cui si ha una predominanza della classe negativa (visita non convertita) e la percentuale dell'accuratezza e della misura F1 mettono in risalto la situazione appena descritta.

La prima osservazione che si può fare è che la maggior parte delle visite (circa il 95.9%) vengono classificate come visite di compratori, mentre il restante 4.1% come visite di non compratori. Questo evidenzia uno dei limiti dell'uso di un dataset sbilanciato, in quanto la classe predominante (visita senza acquisto) è predetta nella maggior parte dei casi.

Come osservato precedentemente, la maggior parte delle visite sono classificate come visite che non portano ad un acquisto, perciò quasi tutti i clienti riceverebbero come contenuti personalizzati quelli scelti per i clienti che non mostrano intenzioni d'acquisto.

Considerando un esempio reale in cui la politica di business adottata preveda che i clienti non propensi ad acquistare dei prodotti durante una visita ricevano un voucher, si avrebbero circa 3.5% casi in cui i clienti non otterrebbero un voucher perché le loro visite sono predette correttamente come visite di compratori. Tuttavia questa soluzione causerebbe l'assegnazione di voucher a quasi tutti i clienti, e causerebbe il regalo erroneo di voucher a circa il 3.5% delle visite (visite predette erroneamente come visite di non compratori, ovvero le visite di clienti che avrebbero comprato anche senza voucher).

Infine, una trascurabile quantità (0.7%) di clienti non compratori ma classificati erroneamente come compratori non usufruirebbero dei contenuti riservati ai clienti non intenzionati ad acquistare.

Capitolo 6

Conclusioni

6.1 Discussioni

Con il presente progetto di tesi si è sviluppato un modello predittivo per la classificazione delle visite dei clienti coprendo diversi aspetti della Data Science. L'obiettivo principale prefissato era quello di riuscire a classificare una visita di un cliente come una visita in cui avverrà un acquisto oppure come una visita senza alcun acquisto.

La fonte dati di partenza ha richiesto una fase iniziale in cui si sono analizzati i dati disponibili e la loro struttura. Successivamente, dopo averli pre-processati, sono stati inseriti in un database a grafo, su cui è stato possibile eseguire direttamente la maggior parte delle operazioni di pre-processing sul dataset da utilizzare nell'algoritmo di machine learning. Durante la fase di salvataggio, oltre a verificare la correttezza della base dati realizzata, sono stati testati diversi algoritmi per poter ottimizzare il caricamento dei dati. Infine sono state scelte le feature più importanti e gli iperparametri dell'algoritmo di classificazione scelto, ovvero l'algoritmo random forest.

Il modello di classificazione ottenuto permette di fare una previsione sull'intenzione d'acquisto di un cliente in base a diversi fattori, tra cui le caratteristiche intrinseche della visita, i dati storici del cliente (visite passate) e le interazioni con il sito web nella visita corrente. Dopo un'accurata analisi del dataset si è scelto di creare il modello predittivo in seguito a un numero definito di azioni eseguite dal cliente, pari a dieci. Questo implica che il modello sia utilizzabile solo dopo che il cliente abbia interagito con il sito web effettuando almeno dieci azioni (visite di pagine del sito, modifica del carrello, ricerca...).

6.1.1 Possibili applicazioni future del modello generato

La figura 6.1 mostra dei possibili scenari in cui il modello creato potrebbe essere utilizzato. Conoscendo le intenzioni del cliente si possono attuare diverse strategie per migliorare l'esperienza d'uso e il tasso di conversione.

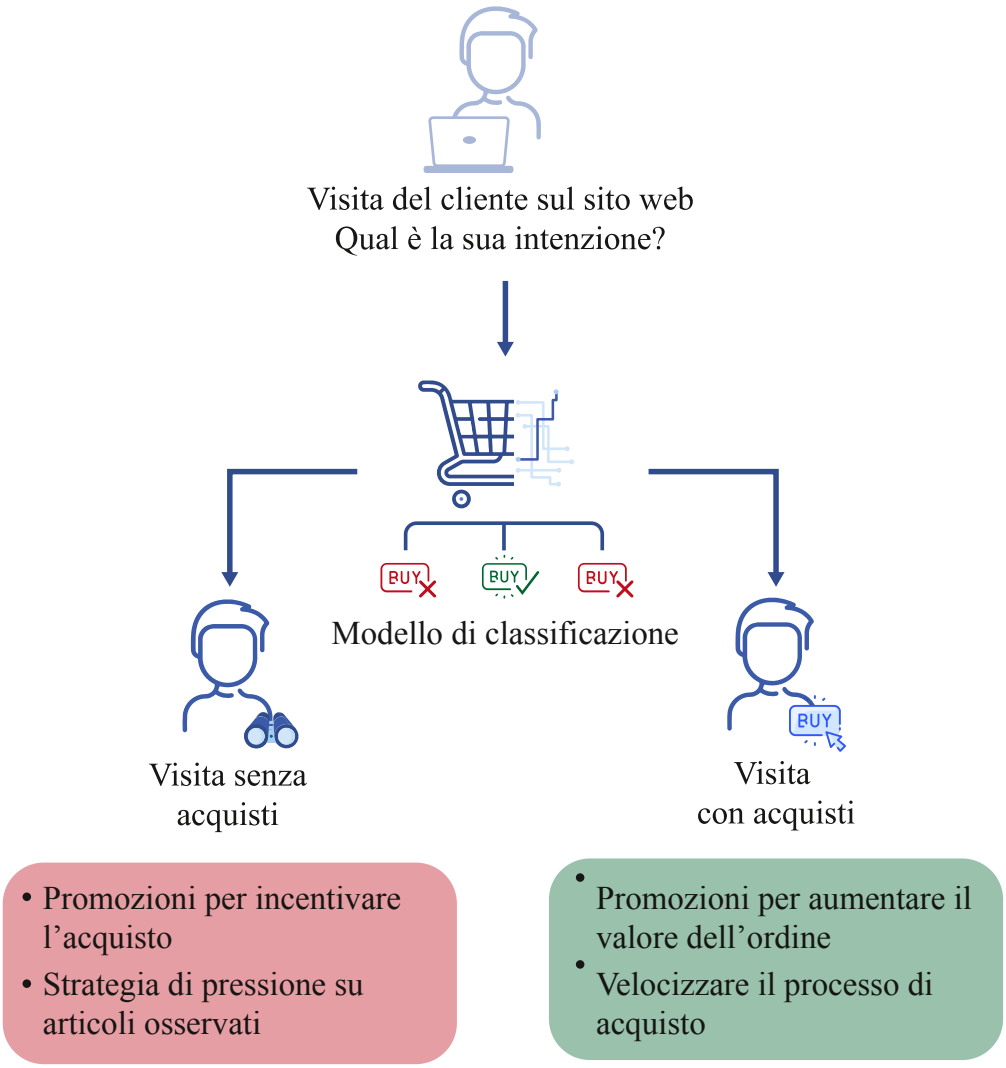

Figura 6.1: Possibili scenari d'utilizzo del modello predittivo

Per esempio si potrebbe assegnare un voucher ai clienti che non mostrano una intenzione d'acquisto oppure si potrebbero guidare più velocemente all'acquisto i clienti per il quale la loro visita viene classificata come una visita di un compratore.

Il modello ottenuto è utilizzabile per poter modificare i contenuti del sito web in modo dinamico, in quanto presenta buoni numeri per poter distinguere diverse visite sul sito web, ma al tempo stesso risente molto della natura del dataset di partenza, in cui la maggior parte delle visite appartiene alla classe relativa alle visite senza acquisto. A causa di questo problema, nel caso in cui si volessero mostrare dei voucher ai clienti non intenzionati ad effettuare alcun acquisito, si attribuirebbero voucher a quasi tutti i clienti (secondo i risultati del capitolo precedente, si assegnano i voucher a circa il 95%). Una possibile soluzione potrebbe essere quella di selezionare tra tutti una percentuale di visite a cui mostrare un voucher e valutarne l'efficacia. L'aspetto positivo nell'utilizzo di questo modello è che si ha comunque un metodo per poter escludere correttamente una buona parte delle visite dei compratori, e quindi non sprecare voucher per clienti che comprerebbero anche senza alcuno sconto.

6.2 Lavori futuri

Nonostante il raggiungimento degli obiettivi prefissati, il modello generato presenta aspetti migliorabili negli sviluppi futuri, nonché la possibilità di poterlo utilizzare per analizzarne l'efficacia. Infatti, uno dei possibili lavori futuri potrebbe essere quello di valutare l'impatto del modello predittivo, applicando una politica di marketing per modificare dinamicamente il contenuto della pagina (come descritto nella sezione 6.1.1).

Un possibile miglioramento del modello può essere ottenuto aggiungendo nuove feature per il training, per esempio utilizzando informazioni relative al cliente provenienti da un'altra fonte dati, oppure migliorando il processo di feature engineering. Nel primo caso si dovrebbe modificare solamente la fase iniziale del progetto di tesi, in cui si esegue il processo ETL.

Come detto precedentemente, il modello generato è utilizzabile non appena il cliente esegue dieci azioni sul sito web. Questa condizione è molto limitante e potrebbe essere uno degli aspetti da migliorare, per esempio con l'utilizzo di altri algoritmi, tra cui per esempio l'utilizzo di catene di Markov per generare il modello nascosto di Markov (Hidden Markov Model). Anche in questo caso il progetto è strutturato in modo tale da permettere questo tipo di miglioramento, in quanto si potrebbe partire direttamente dalla base dati generata. Un altro aspetto importante è che la scelta del database a grafo potrebbe semplificare questo tipo di analisi in quanto le relazioni tra le azioni in una visita sono facilmente accessibili.

Un ulteriore lavoro futuro potrebbe essere quello di generalizzare il processo di salvataggio dei dati e successivamente il processo di pre-processing del dataset per poter eseguire la stessa analisi su diversi siti di e-commerce. In questo modo potrebbe essere interessante anche analizzare l'impatto dei contenuti dinamici in e-commerce B2C rispetto a quelli B2B.

Appendice A

Dataset in input

Le tabelle A.1 e A.2 elencano rispettivamente i parametri dei dataset delle visite e delle azioni. Per ogni attributo è specificato il nome, il tipo, la possibilità o meno di contenere un valore nullo e opzionalmente delle informazioni aggiuntive.

Tabella A.1: Struttura del dataset delle visite

Nome	Tipo	Null
idSite	string	
idVisit	string	
visitIp	string	
visitorId	string	
fingerprint	string	
userId	string	X
actionDetails	array	
goalConversions	int	
siteCurrency	string	
siteCurrencySymbol	string	
serverDate	string	
visitServerHour	string	
lastActionTimestamp	int	
lastActionDateTime	string	
siteName	string	
serverTimestamp	int	

Continua nella prossima pagina

Tabella A.1 - continua dalla pagina precedente

Nome	**Tipo**	**Null**
firstActionTimestamp	int	
serverTimePretty	string	
serverDatePretty	string	
serverDatePrettyFirstAction	string	
serverTimePrettyFirstAction	string	
visitorType	string	
visitorTypeIcon	string	X
visitConverted	string	
visitConvertedIcon	string	X
visitCount	string	
visitEcommerceStatus	string	
visitEcommerceStatusIcon	string	X
daysSinceFirstVisit	int	
secondsSinceFirstVisit	string	
daysSinceLastEcommerceOrder	int	
secondsSinceLastEcommerceOrder	string	X
visitDuration	string	
visitDurationPretty	string	
searches	string	
actions	string	
interactions	string	
referrerType	string	
referrerTypeName	string	
referrerName	string	
referrerKeyword	string	
referrerKeywordPosition	string	X
referrerUrl	string	X
referrerSearchEngineUrl	string	X
referrerSearchEngineIcon	string	X
referrerSocialNetworkUrl	string	X

Continua nella prossima pagina

Tabella A.1 - continua dalla pagina precedente

Nome	**Tipo**	**Null**
referrerSocialNetworkIcon	string	X
languageCode	string	
language	string	
deviceType	string	
deviceTypeIcon	string	
deviceBrand	string	
deviceModel	string	
operatingSystem	string	
operatingSystemName	string	
operatingSystemIcon	string	
operatingSystemCode	string	
operatingSystemVersion	string	
browserFamily	string	
browserFamilyDescription	string	
browser	string	
browserName	string	
browserIcon	string	
browserCode	string	
browserVersion	string	
totalEcommerceRevenue	string	
totalEcommerceConversions	string	
totalEcommerceItems	string	
totalAbandonedCartsRevenue	string	
totalAbandonedCarts	string	
totalAbandonedCartsItems	string	
events	string	
continent	string	
continentCode	string	
country	string	
countryCode	string	

Continua nella prossima pagina

Tabella A.1 - continua dalla pagina precedente

Nome	**Tipo**	**Null**
countryFlag	string	
region	string	X
regionCode	string	
city	string	
location	string	
latitude	string	
longitude	string	
visitLocalTime	string	
visitLocalHour	string	
daysSinceLastVisit	int	
secondsSinceLastVisit	string	
resolution	string	
plugins	string	
pluginsIcons	string	
customVariables	string	
sessionReplayUrl	string	X
campaignId	string	
campaignContent	string	
campaignKeyword	string	
campaignMedium	string	
campaignName	string	
campaignSource	string	
campaignGroup	string	
campaignPlacement	string	
provider	string	X
providerName	string	
providerUrl	string	X

Tabella A.2: Struttura del dataset delle azioni

Nome	**Tipo**	**Null**
type	string	
url	string	X
pageTitle	string	X
pageIdAction	string	X
idpageview	string	X
pageId	string	X
serverTimePretty	string	
timestamp	int	
timeSpent	string	X
timeSpentPretty	string	X
pageviewPosition	string	X
title	string	X
subtitle	string	
icon	string	
iconSVG	string	
productViewCategories	string	X
productViewPrice	string	X
productViewName	string	X
productViewSku	string	X
pageLoadTime	string	X
pageLoadTimeMilliseconds	string	X
siteSearchKeyword	string	X
siteSearchCategory	string	X
siteSearchCount	float	X
eventCategory	string	X
eventAction	string	X
eventName	string	X
eventValue	float	X
revenue	string	X
items	string	X

Continua nella prossima pagina

Tabella A.2 - continua dalla pagina precedente

Nome	**Tipo**	**Null**
itemDetails	array	X
orderId	string	X
revenueSubTotal	string	X
revenueTax	string	X
revenueShipping	float	X
revenueDiscount	float	X

Bibliografia

[1] *E-commerce in Italia 2021*. Rapp. tecn. 2021. URL: https://www.casaleggio.it/wp-content/uploads/2020/12/CA-E-commerce-2021-report-ITA__WEB.pdf (cit. a p. 1).

[2] Darren DeMatas. «10 Types of Ecommerce Business Models That Work Right Now». In: (2021). URL: https://www.ecommerceceo.com/types-of-ecommerce-business-models/ (cit. a p. 5).

[3] D. Court, D. Elzinga, S. Mulder e O. J. Vetvik. «The consumer decision journey». In: (nov. 2009). URL: https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/the-consumer-decision-journey (cit. alle pp. 6, 8).

[4] Katherine N. Lemon e Peter C. Verhoef. «Understanding Customer Experience throughout the Customer Journey». In: 80 (2016), pp. 69, 96. DOI: 10.1509/jm.15.0420 (cit. a p. 8).

[5] Riccardo Mangiaracina, Gianluca Brugnoli e A. Perego. «The eCommerce Customer Journey: A Model to Assess and Compare the User Experience of the eCommerce Websites». In: 14 (dic. 2009) (cit. a p. 9).

[6] Kate Kaplan. «When and How to Create Customer Journey Maps». In: *NN/g Nielsen Norman Group* (lug. 2016). URL: https://www.nngroup.com/articles/customer-journey-mapping/ (cit. alle pp. 10, 11).

[7] Kim Salazar. «7 Ways to Analyze a Customer-Journey Map». In: *NN/g Nielsen Norman Group* (mar. 2020). URL: https://www.nngroup.com/articles/analyze-customer-journey-map/ (cit. a p. 11).

[8] Catherine Vanvonno. *4 Dynamic Content Ideas for Boosting eCommerce Conversions*. 2021. URL: https://understandingecommerce.com/4-dynamic-content-ideas-for-boosting-ecommerce-conversions/ (visitato il 25/07/2021) (cit. a p. 11).

[9] Stephan Serrano. *Complete List of Cart Abandonment Statistics: 2006-2021*. 2021. URL: https://www.barilliance.com/cart-abandonment-rate-statistics/ (visitato il 25/07/2021) (cit. a p. 12).

[10] Komal Helyer. *8 examples of dynamic content that improve the customer experience and boost conversions*. 2020. URL: https://www.pure360.com/8-examples-of-dynamic-content-that-improve-the-customer-experience-and-boost-conversions/ (visitato il 22/07/2021) (cit. a p. 12).

[11] P. Atzeni, S. Ceri, S. Paraboschi e R. Torlone. *Basi di dati. Modelli e linguaggi di interrogazione*. McGraw-Hill Education, giu. 2009 (cit. a p. 13).

[12] E. E Codd. «A relational model for large shared data banks. Communications of the ACM.» In: 13 (1970), pp. 477–387 (cit. a p. 13).

[13] Elena Baralis. *Data science, The Big Data challenge*. URL: https://dbdmg.polito.it/wordpress/wp-content/uploads/2018/10/1-DSIntro.pdf (visitato il 05/10/2021) (cit. a p. 18).

[14] Elena Baralis. *The data mining process*. URL: https://dbdmg.polito.it/wordpress/wp-content/uploads/2018/10/5-DMProcess.pdf (visitato il 05/10/2021) (cit. alle pp. 19, 21).

[15] E. Baralis e S. Chiusano. *Association Rules Fundamentals*. URL: https://dbdmg.polito.it/wordpress/wp-content/uploads/2019/10/DSL-3-MassRules.pdf (visitato il 05/10/2021) (cit. a p. 21).

[16] Elena Baralis. *Classification fundamentals*. URL: https://dbdmg.polito.it/wordpress/wp-content/uploads/2020/11/8-DMClassification.pdf (visitato il 05/10/2021) (cit. alle pp. 22, 23).

[17] Thomas C. Redman. «If Your Data Is Bad, Your Machine Learning Tools Are Useless». In: (apr. 2018). URL: https://hbr.org/2018/04/if-your-data-is-bad-your-machine-learning-tools-are-useless (cit. a p. 27).

[18] Tim Berners-Lee, Larry M Masinter e Roy T. Fielding. *Uniform Resource Identifiers (URI): Generic Syntax*. RFC 2396. Ago. 1998. DOI: 10.17487/RFC2396. URL: https://rfc-editor.org/rfc/rfc2396.txt (cit. a p. 28).

[19] *Reporting API Reference*. URL: https://developer.matomo.org/api-reference/reporting-api (visitato il 27/07/2021) (cit. a p. 33).

[20] *Pandas API reference*. URL: https://pandas.pydata.org/docs/reference/index.html#api (visitato il 25/07/2021) (cit. a p. 33).

[21] Ben Schreck. «Feature Engineering vs Feature Selection». In: (gen. 2018). URL: https://innovation.alteryx.com/feature-engineering-vs-feature-selection/ (cit. alle pp. 58, 60).

[22] Giuseppe Bonaccorso. *Machine Learning Algorithms: A reference guide to popular algorithms for data science and machine learning*. Packt Publishing, lug. 2017 (cit. a p. 61).

[23] *sklearn.ensemble.RandomForestClassifier*. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html` (visitato il 02/10/2021) (cit. a p. 63).

[24] *How does Matomo detect unique and returning visitors? (with user id, visitor id from cookie and/or fingerprint)*. URL: `https://matomo.org/faq/general/faq_21418/` (visitato il 20/07/2021).

[25] CJ Sullivan. «Create a graph database in Neo4j using Python». In: (feb. 2010). URL: `https://towardsdatascience.com/create-a-graph-database-in-neo4j-using-python-4172d40f89c4`.