



Università
Ca' Foscari
Venezia

Corso di Laurea Magistrale
in
Marketing e Comunicazione

Tesi di Laurea

Data Mining e Analisi Predittiva nel Mercato dei Giochi da Tavolo

Relatrice

Ch.ma Prof.ssa Daniela Favaretto

Laureando

Mattia Vignali
Matricola 908698

Anno Accademico

2025 / 2026

Indice

Introduzione	pag. 1
Capitolo 1 – Analisi del Mercato di Riferimento	pag. 3
1.1 – Dimensione e Dinamiche del Mercato Globale.....	pag. 3
1.2 – Effetto della Pandemia Covid-19.....	pag. 4
1.3 – BoardGameGeek Come Infrastruttura Digitale del Settore.....	pag. 5
1.4 – Rilevanza di un’Analisi di Cross-Selling.....	pag. 6
1.5 – Rilevanza di un’Analisi Predittiva.....	pag. 7
Capitolo 2 – Presentazione dei Dataset	pag. 8
2.1 – Il Dataset Giochi.....	pag. 8
2.2 – Il Dataset Recensioni.....	pag. 9
Capitolo 3 – Metodologia di Analisi	pag. 10
3.1 – L’ambiente R e i Pacchetti Utilizzati.....	pag. 10
3.2 – Metodologia per l’Analisi di Cross-Selling.....	pag. 11
3.2.1 – Le Regole di Associazione e l’Algoritmo Apriori.....	pag. 11
3.2.2 – Le Metriche di Valutazione delle Regole: Supporto, Confidenza e Lift.....	pag. 12
3.2.3 – Dalla Recensione al Paniere: la Recensione come Proxy dell’Acquisto.....	pag. 13
3.2.4 – Alberi di Regressione (CART): Principio, Potatura e Criterio 1-SE.....	pag. 13
3.3 – Metodologia per l’Analisi Predittiva.....	pag. 14
3.3.1 – L’albero di Regressione Applicato al Successo dei Giochi.....	pag. 15
3.3.2 – La Random Forest come Estensione dell’Albero di Regressione.....	pag. 15
Capitolo 4 – Analisi di Cross-Selling	pag. 16
4.1 – Pulizia Preliminare dei Dataset.....	pag. 16

4.2 – Controlli di Qualità sui Due Dataset.....	pag. 17
4.2.1 – Controllo di Duplicati Utente-Gioco nel Dataset Recensioni...	pag. 17
4.2.2 – Controllo di Eventuali Duplicati dell'Identificativo id nel Dataset Giochi.....	pag. 17
4.2.3 – Controllo di Eventuali Duplicati di Denominazione dei Giochi.....	pag. 17
4.2.4 – Verifica del Rischio di Distorsione nell'Eventuale Accorpamento.....	pag. 19
4.2.5 – Controllo Valori Mancanti.....	pag. 21
4.2.6 – Controllo del Campo Anno di Pubblicazione nel Dataset Giochi.....	pag. 23
4.2.7 – Controllo del Campo Rating nel Dataset Recensioni.....	pag. 24
4.2.8 – Allineamento dei Due Dataset.....	pag. 24
4.3 – Estrazione delle Regole di Associazione.....	pag. 25
4.3.1 – Costruzione dei Panieri.....	pag. 25
4.3.2 – Selezione dei Parametri e Applicazione dell'Algoritmo Apriori.....	pag. 25
4.4 – Trasformazione delle Regole nel Formato Lungo.....	pag. 29
4.5 – Costruzione dei Predittori per l'Albero di Regressione.....	pag. 29
4.5.1 – Costruzione dell'Indicatore stesso_brand.....	pag. 30
4.5.2 – Costruzione degli Indici di Jaccard.....	pag. 31
4.6 – Matrice di Correlazione tra le Feature e il Lift.....	pag. 33
4.7 – Costruzione dell'Albero di Regressione.....	pag. 34
4.8 – Interpretazione dell'Albero e Individuazione dei Pattern.....	pag. 37
4.8.1 – Componente Strutturale.....	pag. 37
4.8.2 – Affinità di Meccanica.....	pag. 38
4.8.3 – Affinità Residua.....	pag. 39

4.9 – Interpretazione dei Risultati in Ottica di Marketing.....	pag. 40
4.9.1 – Componente Strutturale.....	pag. 40
4.9.2 – Affinità di Meccanica e Residua.....	pag. 41
4.9.3 – Canale e-commerce.....	pag. 41
4.9.4 – Canale Retail Fisico.....	pag. 42
4.9.5 – Implementazione prudente.....	pag. 42
4.10 – Limiti dell’Analisi.....	pag. 43
4.10.1 – Scelta del Dataset e della Soglia Rating.....	pag. 43
4.10.2 – Recensione come Proxy di Acquisto.....	pag. 43
4.10.3 – Concentrazione su Pochi Titoli.....	pag. 43

Capitolo 5 – Analisi Predittiva delle Caratteristiche Associate al Successo dei

Giochi da Tavolo.....	pag. 44
5.1 – Costruzione del Dataset e Trattamento dei Valori Mancanti.....	pag. 44
5.2 – Costruzione dei Predittori.....	pag. 47
5.2.1 – Meccaniche e Categorie.....	pag. 47
5.2.2 – Designer.....	pag. 48
5.2.3 – Trattamento dei Valori Mancanti nel Dataset.....	pag. 49
5.3 – Suddivisione del Dataset in Train/Test su Base Temporale.....	pag. 50
5.4 – Matrice di Correlazione tra i Predittori dell’Albero.....	pag. 51
5.5 – Costruzione e Potatura dell’Albero di Regressione.....	pag. 52
5.5.1 – La Variabile Target: il Geek Rating.....	pag. 52
5.5.2 – Selezione del Minbucket.....	pag. 53
5.5.3 – Costruzione dell’Albero.....	pag. 54
5.5.4 – Valutazione sul Test Set.....	pag. 55
5.6 – Random Forest come Verifica di Robustezza.....	pag. 56
5.6.1 – Costruzione della Random Forest.....	pag. 56

5.6.2 – Valutazione sul Test Set.....	pag. 57
5.6.3 – Confronto dell'Importanza delle Variabili.....	pag. 57
5.7 – Stabilità Temporale delle Associazioni Individuate.....	pag. 59
5.7.1 – Variabili Meccaniche.....	pag. 59
5.7.2 – Variabili Strutturali.....	pag. 60
5.7.3 – Posizione nella Distribuzione del Geek Rating.....	pag. 61
5.8 – Sintesi del Capitolo, Applicazione e Limiti.....	pag. 62
5.8.1 – Sintesi dei Risultati.....	pag. 62
5.8.2 – Applicazione alla Fase di Ideazione del Prodotto.....	pag. 63
5.8.3 – Limiti Ricontrati.....	pag. 64
Capitolo 6 – Conclusioni.....	pag. 66
6.1 – Richiamo ai Risultati.....	pag. 66
6.2 – Complementarietà delle Due Analisi.....	pag. 67
6.3 – Limiti Trasversali.....	pag. 67
6.4 – Sviluppi Futuri.....	pag. 68
Bibliografia.....	pag. 69
Sitografia.....	pag. 71
Appendice.....	pag. 72
Dichiarazione del Ricorso a Strumenti di Intelligenza Artificiale (IA)	pag. 86

Introduzione

Le decisioni di marketing richiedono una conoscenza quanto più approfondita possibile del mercato, dei consumatori e dei prodotti che ne compongono l'offerta. La crescente disponibilità di dati generati dagli utenti attraverso le loro interazioni con prodotti e piattaforme digitali, tra le quali recensioni e cronologie di navigazione e acquisto, offre nuove opportunità di osservazione e analisi, consentendo di comprendere le preferenze dei consumatori, le relazioni tra i prodotti e le caratteristiche che ne influenzano la capacità di attrarre la domanda. Per trasformare tale patrimonio informativo in conoscenza utile ai processi decisionali è necessario ricorrere a strumenti adeguati in grado di estrarre, organizzare e interpretare le informazioni contenute nei dati.

Il presente lavoro prende in esame il mercato dei giochi da tavolo, nel quale, attraverso l'impiego di tecniche di data mining e di analisi predittiva, vengono analizzati i dati con l'obiettivo di identificare pattern di consumo utili alla definizione di strategie di cross-selling e di individuare le caratteristiche maggiormente associate al successo dei giochi.

Per sviluppare gli obiettivi proposti il lavoro è stato articolato in sei capitoli.

Il Capitolo 1 presenta una panoramica del mercato dei giochi da tavolo, discutendone le principali caratteristiche e le ragioni per cui costituisce un caso di studio rilevante.

Il Capitolo 2 descrive i dataset utilizzati nell'analisi, illustrandone la provenienza, il contenuto e le principali caratteristiche.

Il Capitolo 3 illustra la metodologia adottata, descrivendone le procedure di applicazione ai dati e gli strumenti analitici impiegati.

Il capitolo 4 sviluppa l'analisi delle associazioni tra i giochi, individuando pattern di co-apprezzamento utili in un'ottica di cross-selling.

Il capitolo 5 sviluppa l'analisi predittiva delle caratteristiche associate al successo dei giochi da tavolo, verificandone la stabilità su titoli non osservati in fase di stima.

Il capitolo 6, infine, raccoglie le conclusioni del lavoro, discutendone i principali risultati, i limiti dell'analisi e i possibili sviluppi futuri.

Capitolo 1

Analisi del Mercato di Riferimento

Prima di procedere alla presentazione dei dati e della metodologia di analisi, è utile chiarire perché il mercato dei giochi da tavolo costituisca un contesto di analisi rilevante, sia per le sue caratteristiche economiche sia per la disponibilità di dati.

1.1 – Dimensione e Dinamiche del Mercato Globale

Il mercato globale dei giochi da tavolo si colloca, secondo le stime disponibili, in un ordine di grandezza di circa 12 miliardi di dollari nei primi anni Duemilaventi. Naghikhani (2024), nella propria ricerca condotta presso la OCAD University, riporta un valore di 11,88 miliardi di dollari nel 2022 con una traiettoria di crescita costante negli anni successivi. Questa stima trova conferma, nello stesso ordine di grandezza, nei dati pubblicati da Global Market Insights (2025), secondo cui il mercato valeva 12,2 miliardi di dollari nel 2024. Entrambe le fonti concordano inoltre sulla direzione della crescita attesa: le proiezioni indicano un raddoppio del valore di mercato entro la fine del decennio in corso o l'inizio del successivo, con tassi di crescita annua composta che si attestano tra il 5% e il 10% a seconda della fonte.

Il mercato dei giochi da tavolo si presenta fortemente frammentato, infatti secondo Global Market Insights (2025), i primi cinque editori a livello globale – Asmodee, Mattel, Hasbro, Ravensburger e Goliath Games – detengono collettivamente solo il 15-20% del mercato, con il singolo leader Ravensburger fermo al 5%. Con una concentrazione così bassa nessun singolo editore esercita una posizione dominante sul mercato. Inoltre, tra i fattori di crescita del settore, Global Market Insights, indica l'integrazione digitale, la diffusione di *community event* dedicati e una crescente diffusione del crowdfunding per permettere a editori indipendenti di raggiungere il pubblico bypassando i canali editoriali tradizionali.

Questa combinazione di bassa concentrazione, ingresso continuo di nuovi operatori e tassi di crescita previsti del 5-10% annuo composto permette di collocare il mercato dei giochi da tavolo in una fase di crescita.

Sul piano geografico, il Nord America si conferma l'area di maggiore valore assoluto, mentre l'Asia Pacifico risulta la regione a crescita più rapida con un tasso annuo medio composto del 6% nel periodo 2025-2034; Cina, Giappone, Corea del Sud e India vengono segnalati come mercati in forte espansione poiché sostenuti dalla diffusione di locali dedicati e da un pubblico giovanile in crescita. L'Europa, invece, pur crescendo a un ritmo lievemente inferiore (5,7%) mantiene un ruolo centrale sia per i volumi di consumo sia sul piano culturale, in quanto sede storica degli Eurogame (Germania su tutte, con titoli come *Catan* e *Carcassonne*) e di eventi di riferimento per il settore come Essen Spiel (Global Market Insights, 2025).

1.2 – Effetto della pandemia Covid-19

Un aspetto che merita un breve approfondimento riguarda l'effetto della pandemia Covid-19 sulla domanda di giochi da tavolo. Secondo il *NPD Group* nel primo semestre del 2020 la categoria aggregata *Games & Puzzles* è cresciuta del 37% nei dodici mercati internazionali da loro monitorati¹. Un riscontro più mirato sui giochi da tavolo arriva da due dei maggiori editori a livello mondiale: nel 2020 Ravensburger ha registrato un aumento del 22% nelle vendite dei giochi da tavolo (Matalucci, 2021), mentre Hasbro ha riportato una crescita del 15% nella categoria *Gaming*² rispetto al 2019 (Hasbro, 2021). Resta comunque opportuna una certa cautela nell'attribuire questa crescita esclusivamente alla pandemia: come osservato da Ynon Kreiz, CEO di Mattel, la quota di crescita direttamente imputabile al Covid-19 risultava difficile da isolare, dal momento che il comparto

¹ I dodici mercati monitorati da NPD Group sono: Belgio, Brasile, Canada, Francia, Germania, Italia, Messico, Paesi Bassi, Russia, Spagna, Regno Unito e Stati Uniti (NPD Group, 2020).

² La categoria "Gaming" di Hasbro è comprensiva di giochi da tavolo e affini (Hasbro, 2021).

giocattoli³ mostrava già prospettive di espansione prima della pandemia (Ryan, 2021). Nel complesso, le evidenze disponibili suggeriscono quindi che la pandemia possa aver agito soprattutto da acceleratore di una traiettoria di crescita preesistente piuttosto che da causa esclusiva della stessa.

1.3 – BoardGameGeek come Infrastruttura Digitale del Settore

Parallelamente alla crescita del mercato fisico, il settore ha sviluppato una rete di risorse digitali, tra le quali forum, aree di gioco e canali di vendita dedicati, che ne costituisce parte integrante e permette di offrire un punto di aggregazione per gli appassionati e una fonte di dati sul comportamento dei consumatori.

In questo panorama emerge BoardGameGeek, piattaforma online fondata nel 2000 e che nel 2010 ha ricevuto il *Diana Jones Award for Excellent in Gaming*, premio che ne ha riconosciuto il ruolo di risorsa senza pari per gli appassionati di giochi da tavolo e di carte (Diana Jones Award, 2010). La community che si è formata attorno alla piattaforma è ad oggi molto ampia, avendo raggiunto i 4 milioni di utenti nel 2025 (Tabletok Games, 2025), ma non è distribuita in modo uniforme nel mondo, infatti, si tratta in larga parte di utenti nordamericani ed europei, con gli Stati Uniti primi per numero di visite seguiti da Canada, Regno Unito, Australia, Germania e altri paesi europei⁴. Vale la pena tenerlo presente guardando ai risultati che seguiranno poiché le conclusioni saranno legate a questi mercati.

La piattaforma ospita blog e articoli di approfondimento sul settore, e integra un mercato interno, il Geek Market, che consente agli utenti di acquistare, vendere e scambiare direttamente i propri giochi. Questa combinazione di funzioni permette alla piattaforma di essere una vera e propria infrastruttura, in grado di unire in un unico spazio la catalogazione sistematica dei prodotti, la valutazione della community e lo scambio economico tra utenti.

³ Kreiz si riferisce all'intero comparto giocattoli, di cui i giochi da tavolo rappresentano solo una sottocategoria (Ryan, 2021).

⁴ Concentrazione degli utenti per il sito boardgamegeek.com osservata tramite lo strumento online similarweb.com

È proprio per questa mole di dati disponibili che, come si vedrà nel capitolo successivo, da questa piattaforma derivano i dataset su cui poggerà l'intera analisi condotta nel presente lavoro.

1.4 – Rilevanza di un'Analisi di Cross-Selling

Prima di addentrarci nell'analisi vera e propria è utile chiarire cosa si intenda per *cross-selling* e perché si ritiene rilevante analizzarne il possibile impatto nel mercato dei giochi da tavolo.

Il cross-selling è una strategia di vendita che consiste nel proporre a un cliente, già acquirente di un determinato prodotto o servizio, prodotti o servizi aggiuntivi o complementari, con l'obiettivo di ampliare il numero di acquisti dallo stesso cliente presso l'impresa (Kamakura, 2007).

Una delle tecniche analitiche che rendono operativo il cross-selling su larga scala è la *market basket analysis*, introdotta da Agrawal e colleghi nei primi anni Novanta, e che consiste nell'analisi delle associazioni tra prodotti acquistati congiuntamente da uno stesso insieme di consumatori (Agrawal et al., 1993). La logica sottostante è la stessa che sostiene i moderni sistemi di raccomandazione, oggi ben diffusi oltre il solo ambito della vendita al dettaglio, come documentato, ad esempio, dal caso di Netflix: i responsabili del sistema di raccomandazione della piattaforma riportano che tale sistema, già nel 2015, influenzava la scelta di circa l'80% delle ore di visione complessive (Gomez-Uribe & Hunt, 2015). Il contesto è diverso, ma il dato offre un'evidenza dell'impatto che tecniche di questo tipo possono avere.

Il settore dei giochi da tavolo presenta alcune caratteristiche che rendono un'analisi di questo tipo pertinente e con concrete opportunità di applicazione. Infatti, il catalogo di prodotti è ampio e in continua espansione, e orientarsi tra le alternative, sia per chi acquista sia per chi deve decidere cosa proporre, non è sempre semplice. Partire dai comportamenti effettivamente osservati, invece che da giudizi soggettivi o da similarità superficiali tra i prodotti, permette quindi di orientare più concretamente alcune decisioni: dalle raccomandazioni

personalizzate online, alla disposizione dei prodotti a scaffale, fino alla segmentazione della clientela in base a ciò che acquista o apprezza davvero.

Il presente lavoro applica questo approccio nel capitolo 4, dove verranno presentati anche i risultati ottenuti e le possibili implicazioni di marketing.

1.5 – Rilevanza di un'Analisi Predittiva

Oltre all'analisi di cross-selling, il settore dei giochi da tavolo offre un secondo terreno di indagine, volto a comprendere quali caratteristiche di un gioco risultino storicamente associate al suo successo presso la community. A rendere possibile questo tipo di indagine è l'analisi predittiva, intesa come l'insieme di metodi che stimano il valore di una variabile non ancora osservata a partire dai dati storici (Shmueli & Koppius, 2011).

A differenza del cross-selling, che agisce su un prodotto già collocato sul mercato, un'analisi di questo tipo si rivolge idealmente a una fase più a monte, ovvero non un titolo già pubblicato ma un progetto ancora da definire, per il quale individuare quali caratteristiche siano storicamente associate al successo può fornire indicazioni utili. Questo richiede però una cautela interpretativa poiché un'associazione storica osservata sui titoli già pubblicati non equivale automaticamente a un'indicazione operativa per un progetto ancora da realizzare, un punto sul quale il capitolo 5 ritornerà esplicitamente dopo aver condotto l'analisi.

Capitolo 2

Presentazione dei Dataset

Il presente lavoro si basa su due file scaricati a febbraio 2026 dalla piattaforma Kaggle⁵: `games_detailed_info2025.csv` e `bgg-15m-reviews.csv`, entrambi provenienti dal dataset “BoardGameGeek Reviews” pubblicato da Jesse van Elteren nel 2019 e poi aggiornato negli anni successivi. I due file derivano da un'estrazione dei dati di BoardGameGeek (BGG), la piattaforma già presentata nel capitolo 1, effettuata tramite l'API⁶ pubblica del sito.

Poiché si tratta di dati secondari, resi disponibili da soggetti terzi, l'autore non ha potuto esercitare alcun controllo diretto sul processo di estrazione, la cui correttezza dipende interamente da chi lo ha condotto. Un ulteriore limite riguarda più specificatamente la natura di BGG come fonte: trattandosi di un database in continua evoluzione, le modifiche apportate ai dati originali dopo la raccolta non trovano riscontro nei file statici impiegati in questo lavoro.

2.1 – Il Dataset Giochi

Il file `games_detailed_info2025.csv` contiene nella versione originaria 27.780 osservazioni e 52 campi relativi a titoli catalogati su BGG (l'elenco completo è riportato in Appendice, A.2.1). Ogni riga rappresenta un singolo gioco, identificato da un codice numerico (id) e include informazioni descrittive di tipo numerico (anno di pubblicazione, numero minimo e massimo di giocatori, durata media, età minima consigliata), informazioni codificate come liste testuali multi-valore (categoria, meccanica, designer, editore) e metriche aggregate calcolate da BGG sulla base delle valutazioni della community (numero di recensioni, voto medio, voto medio bayesiano, posizionamento in classifica generale e di categoria).

⁵ Kaggle è una piattaforma online di proprietà di Google che ospita dataset pubblici e una comunità di data scientist

⁶ API (Application Programming Interface): insieme di regole che consente a un software di scambiare dati con un altro sistema in modo automatizzato

La selezione dei campi effettivamente utilizzati nelle analisi e i relativi controlli di qualità verranno poi descritti nei successivi capitoli.

2.2 – Il Dataset Recensioni

Il file `bgg-15m-reviews.csv` contiene, nella versione originaria, 15.823.269 osservazioni, ciascuna corrispondente a una singola recensione lasciata da un utente su un gioco. Sulla pagina Kaggle sono disponibili tre versioni del file recensioni, corrispondenti ad aggiornamenti successivi, con rispettivamente circa 15, 19 e 26 milioni di osservazioni. Un confronto sul campo *name* mostra che la versione più recente (quella da 26 milioni di recensioni) copre circa 10.000 titoli unici in più rispetto alla versione qui utilizzata. La scelta di adottare la versione più contenuta è stata motivata dai limiti computazionali riscontrati nelle fasi di pulizia ed elaborazione dei dati in R già con questo volume di osservazioni, che avrebbero reso impraticabile l'utilizzo delle versioni più estese.

I campi principali sono l'identificativo dell'utente, il voto assegnato e l'identificativo del gioco recensito, che consente il collegamento con il dataset giochi. Il file include, inoltre, un campo di commento testuale e un campo con il nome del gioco, oltre a una colonna indice priva di valore informativo ed ereditata dal processo di esportazione, proprio come per il dataset giochi. La selezione dei campi effettivamente utilizzati nell'analisi, e i relativi controlli di qualità, anche in questo caso vengono rimandati ai successivi capitoli.

Capitolo 3

Metodologia di Analisi

Il presente capitolo illustra gli strumenti e i fondamenti teorici che costituiscono l'impianto metodologico impiegato nel presente lavoro, distinguendo la spiegazione generale di ciascun metodo, valida indipendentemente dal dataset a cui viene applicato, dalla sua applicazione specifica ai dati di questo lavoro. Nel corso dell'analisi si ricorre inoltre a tecniche di supporto più circoscritte, per confrontare profili, verificare la significatività o validare scelte, la cui necessità emerge direttamente dai problemi incontrati nel corso dell'analisi stessa. Tali tecniche vengono introdotte contestualmente, all'interno dei capitoli 4 e 5, nel punto in cui il problema che ne motiva l'utilizzo si presenta.

3.1 – L'ambiente R e i Pacchetti Utilizzati.

Entrambe le analisi sono state condotte in R, un ambiente di programmazione e calcolo statistico ampiamente diffuso in ambito accademico e professionale per l'analisi dei dati.

Per l'analisi di cross-selling, sviluppata nel capitolo 4, la mole del dataset recensioni (oltre 15 milioni di osservazioni) ha reso necessario il ricorso al pacchetto *data.table*, le cui prestazioni su volumi di questa scala superano sensibilmente quelle delle strutture dati native di R. L'estrazione delle regole di associazione è stata condotta con il pacchetto *arules* (Hahsler et al., 2005), che implementa l'algoritmo Apriori e le relative metriche di valutazione. L'albero di regressione, costruito a partire dalle regole ottenute, si è basato sul pacchetto *rpart* (Therneau & Atkinson, 1997), con *rpart.plot* per la sua rappresentazione grafica.

Per l'analisi predittiva delle caratteristiche di successo dei giochi da tavolo, sviluppata nel capitolo 5, si è continuato a fare ricorso al pacchetto *data.table* per la manipolazione del dataset, mantenendo così coerenza pur avendo in questo caso un numero di osservazioni decisamente più contenuto (27.780 giochi distinti). Anche qui l'albero di regressione è stato costruito con *rpart* e rappresentato con

rpart.plot. A questo si affianca, per verificarne la robustezza dei risultati, la random forest realizzata utilizzando il pacchetto *ranger* (Wright & Ziegler, 2017).

3.2 – Metodologia per l'Analisi di Cross-Selling

3.2.1 – Le Regole di Associazione e l'Algoritmo Apriori

Le regole di associazione sono un metodo di data mining, introdotto originariamente per l'analisi degli scontrini di vendita al dettaglio (da cui il nome *market basket analysis*; Agrawal et al., 1993) volto a individuare pattern di implicazione del tipo “se A allora B” all'interno di dati transazionali. Apriori (Agrawal & Srikant, 1994) è l'algoritmo classico per l'estrazione efficiente di tali regole e si basa sulla proprietà per cui, se un insieme di prodotti (*itemset*) è frequente, allora anche tutti i suoi sottoinsiemi devono esserlo, di conseguenza un solo sottoinsieme non frequente basta a escludere l'intero *itemset*.

Apriori sfrutta questa proprietà procedendo per passaggi, dai gruppi più piccoli ai più grandi. Si parte contando quante volte compare, nelle transazioni, ciascun prodotto preso singolarmente, e si scartano quelli che non superano la soglia di supporto. Restano quindi solo i prodotti “frequenti”.

A questo punto si formano le coppie, ma solo tra i prodotti già risultati frequenti: se un prodotto è stato scartato al passaggio precedente, nessuna coppia che lo contiene viene nemmeno considerata. Delle coppie così formate si calcola il supporto reale sui dati e si scartano quelle sotto soglia, mantenendo solo le coppie “frequenti”.

Per formare gruppi di tre prodotti, si combinano tra loro le coppie frequenti che condividono un prodotto: ad esempio, se sono risultate frequenti sia la coppia {pane, latte} sia la coppia {pane, uova}, si genera il candidato a tre elementi {pane, latte, uova}. Prima di calcolarne il supporto, si verifica però che anche tutte le sue coppie interne siano frequenti, in questo caso serve quindi che anche {latte, uova} sia frequente, perché se non lo fosse, verrebbe scartato subito senza contarli sui dati. Solo i candidati che superano questo controllo vengono effettivamente

verificati sui dati. Il procedimento si ripete allo stesso modo per i gruppi di quattro elementi e così via, finché non si trovano più gruppi frequenti.

Tutti gli *itemset* risultati frequenti, a qualunque livello del procedimento, vengono infine utilizzati per generare le regole di associazione vere e proprie: per ogni *itemset* frequente, ad esempio {A, B}, si considerano le possibili suddivisioni in antecedente e conseguente (A->B, B->A) e per ciascuna si calcolano le metriche di valutazione discusse nel paragrafo successivo, per decidere se quella specifica regola sia sufficientemente affidabile da essere mantenuta.

Applicato ai dati di questo lavoro, ogni transazione corrisponde al paniere di giochi recensiti da un singolo utente (par. 3.2.3), e ogni regola risultante indica una combinazione di giochi che ricorre nei panieri degli utenti, con una frequenza superiore a quanto specificato dai parametri dell'algoritmo, la cui scelta specifica per questo lavoro è discussa nel capitolo 4 (par. 4.3).

3.2.2 - Le Metriche di Valutazione delle Regole: Supporto, Confidenza e Lift

Una regola di associazione $A \rightarrow B$ ("chi ha A tende ad avere anche B") viene valutata tramite tre metriche complementari: supporto, confidenza e lift (Agrawal et al., 1993).

Il supporto corrisponde alla quota di transazioni (utenti) che contengono sia A che B rispetto al totale, e misura quanto la coppia sia diffusa nel dataset.

La confidenza rappresenta la quota di utenti con A che hanno anche B. Misura l'affidabilità della regola ma non tiene conto della popolarità generale del conseguente B.

Il lift, introdotto da Brin et al. (1997) con il nome originario di *interest*, misura quante volte l'associazione tra A e B sia più (o meno) frequente di quanto atteso in caso di completa indipendenza statistica tra i due eventi. Un valore del lift pari a 1 indica indipendenza, mentre un valore superiore a 1 indica un'associazione positiva e un valore inferiore a 1 un'associazione negativa. A differenza della confidenza, che non tiene conto della popolarità del conseguente, il lift corregge

per la popolarità di entrambi gli eventi ed è per questo che risulta simmetrico e più adatto a individuare associazioni realmente informative piuttosto che semplicemente frequenti.

3.2.3 – Dalla Recensione al Paniere: la Recensione come Proxy dell’Acquisto

L’applicazione di Apriori richiede di ricondurre i dati disponibili alla forma di transazioni, ciascuna composta da un insieme di item. In questo lavoro, ogni transazione corrisponde all’insieme dei giochi recensiti da un singolo utente su BoardGameGeek: un’assunzione che tratta quindi la recensione come proxy dell’acquisto o comunque dell’esperienza di gioco, accogliendo il limite che un utente possa aver recensito un gioco senza averlo prima acquistato.

3.2.4 – Alberi di Regressione (CART): Principio, Potatura e Criterio 1-SE

Gli alberi di regressione (CART; Breiman et al., 1984) sono un metodo di apprendimento supervisionato che suddivide ricorsivamente le osservazioni in sottogruppi (partizionamento ricorsivo), scegliendo a ogni passo la variabile e la soglia di split⁷ che riducono maggiormente, rispetto alla variabile target prescelta, la devianza⁸ interna ai sottogruppi risultanti.

Lasciato crescere senza vincoli l’albero continuerebbe a suddividere le osservazioni fino ad ottenere dei nodi terminali molti piccoli e specifici, che si adattano perfettamente ai dati utilizzati per costruire l’albero ma rischiano di catturarne anche le peculiarità casuali (*overfitting*). Per contenerlo si ricorre alla potatura per costo-complessità (*cost-complexity pruning*), in cui ogni split deve “guadagnarsi” la propria presenza migliorando sensibilmente la capacità predittiva dell’albero, secondo una penalità regolata da un parametro di complessità (cp) che individua una sequenza di alberi via via più semplici.

⁷ Uno split rappresenta la divisione con cui, attraverso la scelta di una variabile e di una soglia, un nodo dell’albero viene separato in due sottogruppi.

⁸ La devianza di un sottogruppo è la somma degli scarti quadratici tra il valore assunto dalla variabile target in ciascuna osservazione del sottogruppo e la media della variabile target nel sottogruppo stesso.

Per scegliere quale di questi alberi generalizzi meglio, si stima l'errore di ciascuno tramite validazione incrociata (*xerror*), una stima comunque soggetta a incertezza campionaria quantificata dal suo errore standard (*xstd*). Per non lasciarsi guidare da differenze dovute al solo rumore campionario, Breiman et al. (1984) propongono il criterio dell'errore standard unico (1-SE): anziché l'albero con *xerror* minimo in assoluto, si seleziona il più semplice tra quelli il cui *xerror* rientra entro una deviazione standard da tale minimo.

In questo lavoro, l'albero non lavora sui dati grezzi delle transazioni, ma su un insieme di feature derivate dalle regole estratte da Apriori, la cui costruzione è discussa nel capitolo 4 (par. 4.5).

3.3 – Metodologia per l'Analisi Predittiva

Prima di adottare l'impianto albero di regressione/random forest descritto nei paragrafi seguenti, era stata inizialmente presa in considerazione l'analisi fattoriale, un metodo consolidato per l'individuazione di variabili statisticamente significative. La sua applicazione è stata però scartata per due motivi legati alla natura dei dati utilizzati: da un lato, alcuni campi del dataset come "meccaniche" e "categorie" presentano valori multi-etichetta (più meccaniche o categorie contemporaneamente), mentre l'analisi fattoriale presuppone relazioni lineari tra le variabili osservate e richiede in ogni caso una trasformazione preventiva dei campi multi-etichetta in variabili a valore singolo; dall'altro, i predittori disponibili sono eterogenei per tipo, includendo variabili continue come la durata, binarie come la presenza o assenza di ciascuna meccanica, e di conteggio come l'esperienza del designer, una combinazione per cui l'analisi fattoriale non è pensata. Gli alberi di regressione e la random forest, discussi nei paragrafi seguenti, gestiscono invece nativamente questa eterogeneità senza richiedere assunzioni di linearità né una struttura di variabili a valore singolo.

3.3.1 – L'albero di Regressione Applicato al Successo dei Giochi

L'albero di regressione impiegato in questa analisi segue lo stesso principio, la stessa logica di potatura e lo stesso criterio 1-SE già descritti nel par. 3.2.4.

Rispetto al caso precedente, qui i predittori sono le caratteristiche dirette dei singoli giochi anziché feature derivate dalle regole di associazione. La variabile target, invece, è una misura del successo del gioco, nello specifico il *Geek Rating* assegnato dalla piattaforma BoardGameGeek, la cui definizione e ragioni della scelta sono discusse nel capitolo 5.

3.3.2 – La Random Forest come Estensione dell'Albero di Regressione

Un singolo albero CART, per quanto interpretabile, soffre di un'elevata varianza: piccole variazioni nei dati di stima possono produrre strutture di split molto diverse, e il problema si accentua con l'aumentare dei predittori candidati. La random forest (Breiman, 2001) affronta il problema costruendo molti alberi, ciascuno su un campione *bootstrap*⁹ dei dati originali e su un sottoinsieme casuale dei predittori a ogni split, mediandone le previsioni e riducendo la varianza.

In questo lavoro la random forest viene impiegata per verificare la classifica di importanza dei predittori ottenuta dall'albero singolo, quella che di fatto guiderà l'interpretazione dei risultati. Questa classifica può risentire della stessa varianza che colpisce l'intera struttura dell'albero, e per questo viene messa alla prova confrontandola con la classifica ottenuta dalla random forest, che rappresenta una misura più stabile perché mediata su centinaia di alberi anziché su uno solo. Se le due classifiche coincidono, l'interpretazione basata sull'albero singolo, più leggibile e per questo mantenuto come strumento primario, trova conferma anche in un metodo meno esposto al caso specifico di una singola esecuzione.

⁹ Un campione bootstrap è ottenuto estraendo osservazioni dal campione originale con reinserimento: il campione risultante ha la stessa dimensione dell'originale ma alcune osservazioni vi compaiono più volte e altre invece non vi compaiono affatto.

Capitolo 4

Analisi di Cross-Selling

Il presente capitolo sviluppa il primo obiettivo del lavoro, enunciato nel capitolo 1: individuare pattern di associazione tra giochi da tavolo utilizzabili in ottica di cross-selling. Il capitolo ripercorre l'intero percorso che ha condotto ai risultati finali: dai controlli di qualità sui due dataset, alla costruzione dei panieri di co-valutazione e all'estrazione delle regole tramite l'algoritmo Apriori, fino alla loro interpretazione attraverso l'albero di regressione costruito su indicatori di similarità tra i giochi coinvolti.

4.1 – Pulizia Preliminare dei Dataset

Nella prima fase di pulizia preliminare è stato scelto di mantenere, all'interno dei due dataset, solo i campi ritenuti utili o potenzialmente tali ai fini dell'analisi.

Codice R: selezione dei campi rilevanti nei due dataset

```
recensioni <- recensioni[, .(user, rating, ID)]
giochi <- giochi[, .(id, name, yearpublished, minplayers, maxplayers,
  playingtime, minage, boardgamecategory, boardgamemechanic,
  boardgamedesigner, boardgamefamily, boardgameexpansion,
  boardgamecompilation, boardgameimplementation, averageweight, usersrated, owned)]
```

Risultato: il dataset recensioni viene ridotto a 3 campi essenziali, mentre il dataset giochi viene ridotto a 17 campi. Tra i campi esclusi figurano i campi testuali estesi come descrizioni, commenti, immagini, thumbnail, ecc.

Un secondo intervento, condotto sui campi testuali del dataset giochi, ha riguardato la codifica dei caratteri: i campi sono stati etichettati esplicitamente come UTF-8 subito dopo la lettura, per correggere errori di decodifica presenti nel dataset originale.

Codice R: correzione dell'*encoding* dei campi testuali

```
campi_testo <- c("name", "boardgamedesigner", "boardgamecategory", "boardgamemechanic",
  "boardgamefamily", "boardgameexpansion", "boardgamecompilation",
  "boardgameimplementation")
for (campo in campi_testo) Encoding(giochi[[campo]]) <- "UTF-8"
```

4.2 – Controlli di Qualità sui Due Dataset

Entrambi i dataset sono stati sottoposti a una serie di controlli di qualità, volti a verificare la presenza di osservazioni duplicate, valori mancanti o anomali, prima di procedere con le fasi successive dell'analisi.

4.2.1 – Controllo di Duplicati Utente-Gioco nel Dataset Recensioni

Nel dataset recensioni ogni riga corrisponde a una singola recensione, e di conseguenza l'identificativo del gioco vi compare, per costruzione, tante volte quante sono le recensioni ricevute. Il seguente controllo di qualità riguarda la verifica della possibile presenza di più recensioni dello stesso utente per lo stesso gioco.

Codice R: verifica dei duplicati sulla coppia user, ID

```
n_dup_user_id <- sum(duplicated(recensioni[, .(user, ID)]))
```

Risultato: 0 osservazioni duplicate, per cui nessun intervento risulta necessario.

4.2.2 – Controllo di Eventuali Duplicati dell'Identificativo id nel Dataset Giochi

Si verifica l'unicità del campo identificativo (*id*), per accertarsi che non vi sia duplicati e che ogni id compaia una sola volta nel dataset.

Codice R: verifica dell'unicità del campo identificativo

```
n_dup_id <- sum(duplicated(giochi$id))
```

Risultato: 0 identificativi duplicati, per cui anche in questo caso nessun intervento è necessario.

4.2.3 – Controllo di Eventuali Duplicati di Denominazione dei Giochi

Il controllo di eventuali nomi duplicati nel dataset giochi assume particolare rilevanza in vista delle fasi successive dell'analisi, poiché se due osservazioni distinte corrispondessero in realtà allo stesso prodotto, le recensioni ad esso relative risulterebbero frazionate tra i due identificativi, andando ad impattare

negativamente sul peso dei giochi nei panieri per Apriori costruiti al par. 4.3. Per effettuare il controllo, il campo “*name*” è stato prima normalizzato andando a rimuovere spazi iniziali e finali, e uniformando le maiuscole e minuscole per evitare falsi negativi dovuti a differenze puramente tipografiche.

Codice R: normalizzazione e conteggio dei nomi duplicati

```
giochi[, name_norm := trimws(tolower(name))]  
conteggio_nomi <- giochi[, .N, by = name_norm][N > 1]
```

Risultato: 474 gruppi¹⁰ di nomi duplicati, per un totale di 1015 osservazioni coinvolte.

La presenza di uno stesso nome non implica necessariamente che si tratti dello stesso prodotto, infatti potrebbe trattarsi di una riedizione oppure di un’omonimia casuale tra prodotti editorialmente distinti. La distinzione è stata analizzata tramite il campo “boardgameimplementation”, con cui BGG dichiara esplicitamente le relazioni di riedizione tra titoli. Poiché il collegamento è dichiarato tipicamente solo dall’edizione più recente verso quella precedente, la verifica è stata condotta a livello di gruppo, cercando il riferimento in una qualsiasi delle righe coinvolte anziché nella singola riga esaminata.

Codice R: verifica del collegamento dichiarato da BGG tra le edizioni di uno stesso gruppo

```
gruppi <- giochi[name_norm %in% conteggio_nomi$name_norm]  
gruppi[, riedizione_confermata := any(sapply(seq_along(name), function(i) {  
  altri_nomi <- name[-i]  
  any(sapply(altri_nomi, function(nm) grepl(nm, boardgameimplementation[i], fixed =  
    TRUE)))  
})), by = name_norm]
```

Risultato: 123 dei 474 gruppi risultano confermati come riedizione dichiarata da BGG, mentre i restanti 351 non presentano alcun riferimento incrociato dichiarato dalla fonte.

¹⁰ Il termine gruppo indica l’insieme dei giochi che condividono lo stesso nome normalizzato. Nella maggior parte dei casi (421 dei 474 gruppi) è composto da due sole osservazioni, ma nei rimanenti ne comprende tre o più, fino a un massimo di sei.

Per i 351 gruppi non confermati si è scelto di non procedere ad alcun accorpamento. Questo perché l'unione di due identificativi distinti costituisce un'affermazione empirica, "si tratta dello stesso prodotto", che richiede una prova a supporto, in assenza della quale, il criterio più prudente risulta essere il mantenimento dello status quo dei dati. A titolo puramente illustrativo, uno dei 351 gruppi non confermati (*Zen Garden*, edizioni 2013 e 2020) condivide denominazione e ambientazione tematica ma presenta editore e meccaniche distinti – un esempio concreto del tipo di omonimia non riedizione che il criterio adottato lascia correttamente separata.

Per i 123 gruppi confermati come riedizione, invece, la decisione sull'eventuale accorpamento non è immediata, e verrà verificata di seguito.

4.2.4 – Verifica del Rischio di Distorsione nell'Eventuale Accorpamento

Il fatto che, per 123 gruppi, due identificativi siano stati confermati come la stessa opera editoriale (par. 4.2.3) non implica automaticamente che i loro recensori debbano essere trattati, ai fini dell'analisi, come un unico pubblico. Il paragrafo precedente risponde a una domanda catalografica – è lo stesso prodotto? – ma l'aspetto rilevante per la costruzione dei panieri di *co-apprezzamento* (par. 4.3) non è il prodotto in sé, ma è il comportamento di chi lo recensisce e in particolare quali altri giochi tende a recensire insieme ad esso.

Se i recensori delle due edizioni di uno stesso gruppo condividessero pattern di co-associazione praticamente identici, accorpare i due identificativi sarebbe innocuo poiché si sommerebbero due fonti dello stesso segnale. Ma se i due pubblici avessero invece gusti sistematicamente diversi, l'accorpamento andrebbe a mescolare due segnali distinti in uno solo, facendo sì che il gioco unificato risulterebbe associato sia ai titoli tipici del primo pubblico sia a quelli del secondo. Prima di accorpare i 123 gruppi confermati è quindi necessario verificare quale dei due scenari si presenta nei dati.

A questo scopo è stato condotto un test sugli otto gruppi confermati con il maggior numero di recensioni complessive¹¹. All'interno di ciascun gruppo, l'edizione con il maggior numero di recensioni è stata assunta come edizione di riferimento, e l'altra come edizione minoritaria¹². Per ciascuna delle due popolazioni di recensori sono stati individuati i dieci giochi, esterni al gruppo, che essa tende a recensire più di frequente. Il grado di sovrapposizione tra le due liste di dieci titoli così ottenute, una per edizione, è stato misurato tramite l'indice di Jaccard¹³ (Jaccard, 1912).

Poiché, come si vedrà nel paragrafo dedicato alla costruzione dei panieri (par. 4.3), l'analisi adotterà la soglia $\text{rating} \geq 8$ (andando a ridurre il numero di recensioni), il test è stato eseguito due volte: una prima volta sull'intero storico di recensioni di ciascun utente, e una seconda volta restringendo il numero di recensioni a quelle con $\text{rating} \geq 8$. Per poter rendere le due versioni del test confrontabili sono stati mantenuti gli stessi 8 gruppi in entrambi i test, questo perché se gli otto gruppi cambiassero da una versione all'altra, un'eventuale differenza nei risultati non sarebbe più attribuibile alla soglia di voto, ma al semplice cambio dei casi osservati.

Calcolato sull'intero storico di recensioni, l'indice di Jaccard, riportato di seguito nella tabella 4.1, si colloca tra lo 0,25 e lo 0,67 per sei degli otto gruppi testati, i restanti due (*Dominion* e *Bohnanza*) restano non applicabili per l'assenza di recensori sull'edizione minoritaria nel dataset originale. Il quadro non cambia sostanzialmente restringendo il test alle sole recensioni con $\text{rating} \geq 8$: l'indice si mantiene nello stesso ordine di grandezza per la maggior parte dei gruppi, con la

¹¹ I gruppi sono stati ordinati per numero complessivo di recensioni presenti nel dataset recensioni. Sono stati selezionati i primi otto poiché ritenuti un numero sufficientemente ampio da garantire un campione stabile, mantenendo al contempo una tabella di risultati leggibile.

¹² Nei due degli otto gruppi scelti con più di due edizioni (*Coup* e *Cosmic Encounter*) tutte le edizioni diverse da quella di riferimento (la più recensita) sono state trattate come un'unica popolazione minoritaria.

¹³ L'indice di Jaccard è il rapporto tra la cardinalità dell'intersezione e quella dell'unione di due insiemi. Assume 0 quando i due insiemi non hanno elementi in comune, 1 quando coincidono esattamente. Il codice completo del test eseguito è riportato in Appendice, cod. A.4.1.

sola eccezione di *Santorini*, che scende a 0, mentre i casi non applicabili rimangono tali.

Gruppo	Jaccard (Rec. Totali)	Jaccard (Rec. Rating≥8)
<i>Dominion</i>	0,00 (n.a)	0,00 (n.a)
<i>Love Letter</i>	0,43	0,33
<i>Citadels</i>	0,25	0,18
<i>Coup</i>	0,25	0,18
<i>Bohnanza</i>	0,00 (n.a)	0,00 (n.a)
<i>Arkham Horror</i>	0,67	0,54
<i>Santorini</i>	0,25	0,00
<i>Cosmic Encounter</i>	0,43	0,43

Tabella 4.1 – Indice di Jaccard tra i profili di co-associazione delle due edizioni, per gruppo confermato, su recensioni totali e su rating≥8. Tabella prodotta sulla base dei risultati ottenuti su R.

In nessun caso l'indice di Jaccard si avvicina al valore massimo teorico di 1 (liste interamente coincidenti): anche nel caso più alto osservato (*Arkham Horror*, 0,67), un terzo dei titoli complessivamente associati alle due edizioni non risulta condiviso tra le due liste. I due pubblici risultano quindi più diversi che simili in entrambe le versioni del test, e per questo motivo, ritenendo l'accorpamento rischioso, si è deciso di non accorpare i gruppi confermati.

Pur non modificando la conclusione del test, va segnalato un limite relativo alla numerosità campionaria, con l'applicazione del filtro rating≥8, l'edizione minoritaria coinvolge popolazioni ridotte per alcuni gruppi (18 utenti per *Coup*, 126 per *Santorini*), il che rende l'indice più sensibile a variazioni casuali rispetto alla versione calcolata sull'intero storico di recensioni.

4.2.5 – Controllo Valori Mancanti

Si verifica ora la presenza di valori mancanti in entrambi i dataset, per individuare eventuali campi che richiedano una correzione o una gestione specifica prima delle fasi successive dell'analisi.

Codice R: controllo valori mancanti in entrambi i dataset

```
colSums(is.na(giochi))  
colSums(is.na(recensioni))
```

Risultato: nessun valore mancante rilevato in nessuno dei campi mantenuti per entrambi i dataset.

Un controllo più attento ha però permesso di evidenziare, per quanto riguarda il dataset giochi, un'assenza di informazione per diversi campi, dovuta alla presenza di stringa vuota o liste contenenti solo segnaposto¹⁴. La tabella 4.2 seguente riporta per i diversi campi la condizione di assenza informativa controllata e il numero di osservazioni che vi ricadono.

Campo	Condizione	N. osservazioni
<i>boardgamedesigner</i>	nessun designer accreditato	2.620
<i>boardgamemechanic</i>	lista vuota	1.765
<i>boardgamecategory</i>	lista vuota	372
<i>boardgamefamily</i>	lista vuota	3.892
<i>minage</i>	= 0	1.588
<i>playingtime</i>	= 0	1.173
<i>minplayers</i>	= 0	51
<i>maxplayers</i>	= 0	174
<i>maxplayers</i>	≥ 99	325
<i>averageweight</i>	= 0	494

Tabella 4.2 – Numero di osservazioni con condizione di informazione mancante per diversi campi del dataset giochi. Tabella prodotta sulla base dei risultati ottenuti su R.

I valori segnaposto possono influenzare fasi differenti dell'analisi e vengono pertanto affrontati nel punto in cui producono un effetto concreto, anziché in questa fase preliminare.

¹⁴ Il codice del controllo effettuato è riportato in Appendice, cod. A.4.2.

Per i campi lista (designer, meccaniche e categorie), la gestione è rinviata alla costruzione dell'indice di Jaccard al par. 4.5.2 – stesso indice già introdotto al par. 4.2.4, qui applicato al confronto tra le caratteristiche dichiarate di due giochi anziché tra liste di titoli – tramite una funzione che restituirà NA quando almeno uno dei due insiemi confrontati è privo di informazioni, distinguendo così l'assenza di dati dall'assenza di sovrapposizione. Per i campi numerici, invece, la gestione viene rinviata alle analisi descrittive e ai confronti tra i pattern risultanti, dove i valori segnaposto saranno esplicitamente esclusi dai calcoli, evitando che vengano interpretati come osservazioni valide.

4.2.6 – Controllo del Campo Anno di Pubblicazione nel Dataset Giochi

Si verifica la presenza di valori anomali nel campo anno di pubblicazione (*yearpublished*) del dataset giochi.

Codice R: verifica dei valori presenti nel campo anno di pubblicazione

```
min(giochi$yearpublished)
max(giochi$yearpublished)
sum(giochi$yearpublished == 0)
```

Risultato: l'anno minimo osservato è -3500, quello massimo 2026. 242 giochi con anno pari a 0.

L'anno minimo osservato corrisponde a *Senet*, un gioco dell'antico Egitto; tra i valori più bassi figurano anche giochi tradizionali come le *Biglie* (-3000), *Go* (-2200) e *Tris* (-1300), a cui BGG assegna un anno di origine storica anziché una data di pubblicazione commerciale. Si tratta di titoli privi di un editore e di un vero e proprio lancio sul mercato, e quindi concettualmente estranei alla categoria di giochi da tavolo moderno a cui è rivolta questa analisi.

Per escludere questi casi, insieme ai 242 titoli con anno pari a 0 per cui il dato è semplicemente assente, è stato applicato un filtro che impone 1950 come anno minimo – valore indicativo, scelto per delimitare convenzionalmente l'insieme dei giochi da tavolo moderni rispetto ai giochi tradizionali.

Codice R: filtro anno minimo

```
giochi <- giochi[yearpublished >= 1950]
```

Risultato: il dataset giochi risultante conta ora 27.303 osservazioni.

4.2.7 – Controllo del Campo Rating nel Dataset Recensioni

Si verifica ora che i voti relativi alle recensioni rilasciate dagli utenti rientrino all'interno del range 1-10 dichiarato da BGG.

Codice R: verifica del range di voti

```
sum(recensioni$rating < 1) # voti troppo bassi  
sum(recensioni$rating > 10) # voti troppo alti
```

Risultato: 13 osservazioni hanno presentato un voto inferiore a 1 mentre nessuna superiore a 10.

Sebbene queste osservazioni non avrebbero comunque influito sul risultato finale, poiché come si vedrà nel paragrafo 4.3 verrà applicata la soglia $\text{rating} \geq 8$, si è deciso di rimuoverle ugualmente per mantenere pulito il dataset di lavoro.

Codice R: impostazione del corretto range di voti

```
recensioni <- recensioni[rating >= 1 & rating <= 10]
```

4.2.8 – Allineamento dei Due Dataset

A conclusione dei controlli precedenti, si procede all'allineamento dei due dataset, rimuovendo dal dataset recensioni le osservazioni relative a giochi non più presenti nel dataset giochi.

Codice R: allineamento dei dataset

```
n_non_allineate <- sum(!recensioni$ID %in% giochi$id)  
recensioni <- recensioni[ID %in% giochi$id]
```

Risultato: 345.872 recensioni relative a identificativi non più presenti nel dataset giochi (esclusi dal filtro sull'anno di pubblicazione, par. 4.2.6) sono state rimosse. Il dataset recensioni risultante conta 15.477.384 osservazioni.

4.3 – Estrazione delle Regole di Associazione

4.3.1 – Costruzione dei Panieri

Le recensioni sono state raggruppate per utente in panieri, ciascuno costituito dall'insieme dei giochi recensiti. Si verifica quindi la distribuzione dimensionale dei panieri, condotta sull'intero storico di recensioni.

Codice R: costruzione dei panieri e verifica dimensionale

```
transazioni <- recensioni[, .(items = list(as.character(ID))), by = user]
summary(lengths(transazioni$items))
p995 <- quantile(lengths(transazioni$items), 0.995)
sum(lengths(transazioni$items) > p995)
```

Risultato: la mediana dei panieri è di 12 giochi per utente, un numero contenuto a fronte di un massimo che arriva a 5.944. Questa distanza enorme è determinata da una minoranza molto ristretta: solo 1.749 utenti, lo 0,5% del totale, superano il 99,5° percentile fissato a 641 giochi, producendo i panieri più estesi dell'intero dataset.

4.3.2 – Selezione dei Parametri e Applicazione dell'Algoritmo Apriori

Un primo tentativo di esecuzione diretta dell'algoritmo Apriori, senza applicare alcuna scrematura preventiva sugli utenti e con un supporto minimo (10%) deliberatamente elevato, è stato condotto per verificare la fattibilità computazionale.

Codice R: primo tentativo con supporto elevato

```
transazioni_arules <- as(transazioni$items, "transactions")
apriori(transazioni_arules,
parameter = list(supp = 0.1, conf = 0.6, minlen = 2, maxlen = 3))
```

Risultato: 51 regole ottenute. Solo 38 dei 18.876 giochi distinti superano la soglia di supporto, e sono quindi gli unici ad entrare nel calcolo delle regole. Il numero ridotto di regole è la conseguenza di una soglia di supporto elevata: un'analisi su soli 38 titoli rischierebbe di limitarsi alla conferma di associazioni già note tra i prodotti più diffusi.

Un secondo tentativo è stato condotto per verificare se, riducendo il supporto minimo, fosse possibile ampliare il numero di giochi coinvolti e di regole ottenute rispetto al tentativo precedente.

Codice R: secondo tentativo con supporto al 5% e confidenza mantenuta invariata.

```
apriori(transazioni_arules,  
parameter = list(supp = 0.05, conf = 0.6, minlen = 2, maxlen = 3))
```

Risultato: 1.715 regole, con lift compreso tra 2,20 e 5,04 (mediana 3,12).

Codice R: terzo tentativo con un supporto ridotto al 1% e confidenza mantenuta invariata.

```
apriori(transazioni_arules,  
parameter = list(supp = 0.01, conf = 0.6, minlen = 2, maxlen = 3))
```

Risultato: 453.095 regole generate, con lift compreso tra 2,18 e 31,13 (mediana 4,63).

Vedendo questi risultati, e non essendo gestibile un insieme di 453.095 regole, si è scelto di portare avanti il caso con supporto al 5%.

Parallelamente alla scelta del supporto, si è valutato se costruire i panieri sull'insieme di tutte le recensioni, indipendentemente dal voto assegnato, oppure se limitarli alle recensioni con voto elevato. Nel primo caso i panieri riflettono la *co-esposizione* tra giochi, cioè il semplice fatto che uno stesso utente abbia recensito entrambi i titoli, a prescindere dal giudizio espresso. Restringendo invece l'attenzione ai soli voti alti, i panieri riflettono il *co-apprezzamento*, ovvero non la semplice esposizione a entrambi i titoli ma un giudizio positivo condiviso su entrambi – condizione più pertinente per un'analisi di cross-selling, che punta a individuare cosa viene realmente apprezzato insieme, non solo cosa viene recensito insieme.

Codice R: distribuzione dei voti e quota di recensioni per soglia

```
summary(recensioni$rating)  
mean(recensioni$rating >= 6)  
mean(recensioni$rating >= 8)
```

Risultato: l'85,9% delle recensioni ha voto ≥ 6 , il 37,6% voto ≥ 8 . Questo conferma che le community di appassionati votano sistematicamente nella parte alta della scala, e un filtro rating ≥ 6 esclude solo la coda più severa senza isolare un giudizio di reale apprezzamento.

Un confronto empirico¹⁵, a parità di soglia di supporto (1%) e confidenza (0,6), tra il sottoinsieme rating ≥ 6 e il sottoinsieme rating ≥ 8 ha mostrato 190.364 regole nel primo caso contro 324 nel secondo. Il confronto tra le distribuzioni del lift chiarisce la natura di questa differenza: a rating ≥ 6 il lift minimo resta contenuto, 2,30 con mediana 4,40, mentre a rating ≥ 8 il lift sale raggiungendo 4,05 con una mediana di 5,44. Tra i due, quindi, risulta preferibile il filtro rating ≥ 8 .

Confrontando questo risultato con il riferimento portato avanti in precedenza (supp 0,05/conf 0,6 sull'intero storico di recensioni), il sottoinsieme rating ≥ 8 produce un insieme di regole più contenuto con lift sistematicamente più elevato su tutta la distribuzione, come mostrato nella tabella 4.3 sottostante. È stato pertanto adottato il filtro rating ≥ 8 (339.056 utenti totali, 18.625 giochi distinti), con supporto 0,01 e confidenza 0,6.

	Apriori su tutte le recensioni	Apriori con applicazione filtro rating ≥ 8
Regole	1.715	324
Lift minimo	2,20	4,05
Lift mediana	3,12	5,44
Lift massimo	5,04	10,55

Tab. 4.3 – Confronto tra l'applicazione dell'algoritmo Apriori su tutte le recensioni, con supporto 0,05, e sulle recensioni dopo il filtro sul rating, con supporto 0,01, a parità di confidenza (0,6). Tabella prodotta sulla base dei risultati ottenuti su R.

¹⁵ Il codice completo è riportato in Appendice, cod. A.4.3.

Infine, per verificare la stabilità della combinazione adottata, sono stati esplorati alcuni valori limitrofi di supporto e confidenza¹⁶ (risultati riportati nella tabella 4.4).

Supporto	Conf. 0,5	Conf. 0,6	Conf. 0,7
0,010	1.054	324	77
0,011	733	233	57
0,013	360	103	26

Tab. 4.4 – Numero di regole ottenute al variare di supporto e confidenza dopo l'applicazione del filtro $rating \geq 8$. Tabella prodotta sulla base dei risultati ottenuti su R.

Sul fronte del supporto, passando da 0,011 (233 regole) a 0,010 (324 regole) a parità di confidenza, il lift minimo e massimo restano identici (4,05 e 10,55) e la mediana varia in modo trascurabile (5,40 contro 5,44): le 91 regole aggiuntive incluse dalla soglia più permissiva non risultano quindi più deboli delle altre. Sul fronte della confidenza, passando da 0,5 (1.054 regole) a 0,6 (324 regole) a parità di supporto, il lift minimo sale invece in modo netto da 3,31 a 4,05: la confidenza più alta esclude proprio le regole più deboli, a fronte di una perdita più consistente in termini di numerosità.

Sulla base di quanto emerso, il set di regole definitivo è stato estratto applicando la combinazione prescelta supporto 0,010 e confidenza 0,6.

Codice R: estrazione finale delle regole

```
trans_8 <- recensioni[rating >= 8, .(items = list(as.character(ID))), by = user]
regole <- apriori(as(trans_8$items, "transactions"),
  parameter = list(supp = 0.010, conf = 0.6, minlen = 2, maxlen = 3))
regole <- regole[quality(regole)$lift > 1]
regole <- regole[!is.redundant(regole)]
```

Risultato: 324 regole, lift compreso tra 4,05 e 10,55 (mediana 5,44); nessuna ridondanza tra le regole rilevata.

¹⁶ Il codice per l'esplorazione dei valori limitrofi è riportato in appendice, cod. A.4.4

4.4 – Trasformazione delle Regole nel Formato Lungo

Le 324 regole estratte si presentano nel formato standard antecedente→conseguente, corredato dai rispettivi indicatori di supporto, confidenza e lift. Una regola è, ad esempio: *Le Havre* → *Agricola* (supporto 0,029, confidenza 0,62 e lift 5,16).

Si è ritenuto opportuno trasformare le regole nel formato lungo, scomponendole su più righe – una per ciascun titolo che compone la regola – così da poter unire a ciascuna riga le caratteristiche del gioco corrispondente (meccaniche, categorie, designer, ecc.) e rendere possibile il calcolo dei *predittori* da utilizzare nell'albero di regressione¹⁷.

Con la trasformazione si è osservato che, delle 324 regole, 20 sono a singolo antecedente e 304 a doppio antecedente, per un totale di 952 righe nel formato lungo. Inoltre, è stato osservato che nessuna regola presenta più di un conseguente e che tutti gli identificativi trovano corrispondenza nel dataset giochi.

4.5 – Costruzione dei Predittori per l'Albero di Regressione

Per poter interpretare le regole tramite un albero di regressione servono un target, e come già anticipato, un insieme di *predittori* calcolati per ciascuna riga del formato lungo. Come si vedrà più avanti, il target scelto è il lift della regola.

Nel formato lungo sono state riportate per ogni gioco le caratteristiche ritenute utilizzabili come *predittori*: meccaniche, categorie e designer. Tra le caratteristiche riportate figura anche la complessità (*averageweight*) del gioco, che a differenza delle precedenti, verrà utilizzata in seguito. Le prime tre, così come sono, non possono però essere utilizzate direttamente poiché sono campi lista o testuali, non valori numerici, e un albero di regressione richiede *predittori* numerici su cui operare gli split. Per questo motivo sono stati costruiti sei indici di similarità di

¹⁷ Il codice completo per la trasformazione nel formato lungo è riportato in Appendice, cod. A.4.5.

Jaccard, calcolati separatamente su meccaniche, categorie e designer, nella versione massima e minima per tenere conto delle regole a doppio antecedente.

Un settimo *predittore*, denominato *stesso_brand*, non deriva invece dalle caratteristiche dei giochi appena illustrate: è un indicatore binario, costruito a partire da un insieme distinto di informazioni dichiarate da BGG sulle relazioni tra prodotti – osservabili nel dataset originale giochi – che segnala se antecedente e conseguente appartengono alla stessa famiglia, essendo uno la variante o espansione dell'altro.

Si parte dall'osservare la costruzione di quest'ultimo indicatore.

4.5.1 – Costruzione dell'Indicatore *stesso_brand*

L'indicatore *stesso_brand* serve a distinguere, come anticipato, quali tra le 324 regole riflettono un legame puramente strutturale tra due prodotti (un gioco e la sua stessa espansione, due edizioni dello stesso titolo, due giochi della stessa serie) da quelle che riflettono invece un vero pattern di gusto condiviso tra prodotti indipendenti – ritenuto più rilevante per i fini dell'analisi.

Per costruire questo indicatore sono stati verificati quattro criteri, riconducibili a due gruppi:

1. **Tag di famiglia editoriale** -> il campo *boardgamefamily* di BGG include diversi tag (numero di giocatori, paese di origine, componenti fisici, ecc.), tra cui quello con prefisso "*Game:*", dedicato specificatamente alle relazioni di franchise o serie (ad esempio "*Game: Catan*", condiviso da tutti i titoli della serie *Catan*).
2. **Relazioni dichiarate esplicitamente da BGG** -> tre campi strutturati, *boardgameexpansion*, *boardgamecompilation* e *boardgameimplementation*, permettono a BGG di dichiarare che un gioco è l'espansione, la compilation o l'implementazione di un altro. Il controllo verifica, per ciascuno di questi tre campi, se il nome del conseguente compare nella lista dichiarata dall'antecedente, o viceversa – in entrambe le direzioni, perché come

osservato al par. 4.2.4 questi collegamenti sono spesso dichiarati da una sola delle due parti.

È stato inoltre verificato se qualcuno dei 68 giochi coinvolti nelle 324 regole appartenga a uno dei 474 gruppi di nomi duplicati individuati al par. 4.2.3: solo 2 vi appartengono ma non compaiono mai nella stessa regola, non rendendo necessario intervenire¹⁸.

Il tag di famiglia intercetta da solo 31 delle 324 regole, mentre i tre campi strutturati ne intercettano complessivamente 16, di cui solo 4 non catturate dal tag di famiglia. In totale, l'indicatore *stesso_brand* assume valore 1 per 35 delle 324 regole.

4.5.2 – Costruzione degli Indici di Jaccard

Costruito l'indicatore *stesso_brand*, ci si concentra ora sul calcolo degli indici di Jaccard per le restanti caratteristiche da utilizzare come *predittori*: meccaniche, categorie e designer.

Attraverso le stesse funzioni già utilizzate al par. 4.2.5¹⁹, è stato verificato che nessuno dei 68 giochi coinvolti nelle 324 regole presenta, per i campi designer, meccaniche e categorie, i valori segnaposto riscontrati nell'analisi iniziale: la gestione di questi casi come dati mancanti non trova quindi applicazione su questo campione. Resta invece necessaria l'estrazione dei valori dai campi lista, poiché meccaniche, categorie e designer sono memorizzati da BGG come stringa testuale (ad esempio “[*Uwe Rosenberg*]”) – l'operazione viene svolta prima del calcolo degli indici di Jaccard per ciascuna regola.

Per ciascuna coppia antecedente-consequente, l'indice di Jaccard (già definito al par. 4.2.4) misura quanto le caratteristiche dei due giochi si sovrappongono: si

¹⁸ Il codice completo per la costruzione dell'indicatore *stesso_brand* è riportato in Appendice, cod. A.4.6.

¹⁹ Le funzioni sono due: *parse_designer()* e *parse_lista()*, e vengono utilizzate nel codice riportato in Appendice cod. A.4.7.

prende l'insieme delle meccaniche (o delle categorie, o dei designer) dichiarate per ciascuno dei due, e si calcola quanti elementi hanno in comune rispetto al totale degli elementi complessivamente coinvolti. Un valore vicino a 1 significa che i due giochi coinvolti condividono quasi tutte le meccaniche (o categorie, o designer), mentre un valore vicino a 0 che non ne condividono quasi nessuna.

L'indice viene calcolato separatamente per ciascuna delle tre caratteristiche e, all'interno di ciascuna, separatamente per ogni antecedente della regola rispetto al conseguente – mai tra gli antecedenti stessi, poiché l'obiettivo è misurare quanto un titolo già noto anticipi quello suggerito, non la coerenza interna tra i titoli di partenza - producendo così un valore per le regole a singolo antecedente, due per quelle a doppio antecedente. Di questi valori si conservano sempre il massimo e il minimo, indipendentemente dal numero di antecedenti: per le regole a singolo antecedente, dove esiste un solo confronto possibile, i due coincidono banalmente; per quelle a doppio antecedente rappresentano invece il confronto più e meno favorevole tra gli antecedenti e il conseguente. La scelta di conservare massimo e minimo evita di mescolare in un unico valore situazioni molto diverse tra loro: una regola in cui un antecedente è molto simile al conseguente e l'altro per nulla produrrebbe, con la media, un valore intermedio che non rappresenta bene nessuno dei due casi. Massimo e minimo, mantenuti distinti, permettono invece all'albero di individuare eventuali split basati specificamente sul confronto più favorevole o su quello meno favorevole, un'informazione che la media perderebbe. Applicare la stessa logica a entrambi i casi garantisce che tutte le 324 regole condividano lo stesso insieme di predittori numerici (`mecc_max/min`, `cat_max/min`, `designer_max/min`), necessario per costruire un unico albero di regressione su tutte le regole insieme.

Il dataset²⁰ finale composto da una riga per regola, pronto per l'analisi di regressione conta 324 righe e 11 colonne, tra cui: `lift`, `supporto`, `confidenza`,

²⁰ Il codice completo per la costruzione del dataset finale è riportato in Appendice, cod. A.4.8.

stesso_brand e i sei indici di Jaccard precedentemente definiti. Un controllo ha, inoltre, confermato che tutti gli indici sono compresi nell'intervallo [0,1], e che nessuno presenta un valore mancante.

4.6 – Matrice di Correlazione tra le Feature e il Lift

Prima di procedere alla costruzione dell'albero di regressione, è stata verificata la correlazione tra le sei *feature* di similarità di Jaccard, l'indicatore *stesso_brand* e il lift, al fine di interpretare correttamente gli split generati con l'albero di regressione.

Codice R: matrice di correlazione di Pearson

```
regole_albero[, stesso_brand_num := as.numeric(stesso_brand)]
variabili_corr <- regole_albero[, .(mecc_max, mecc_min, cat_max, cat_min, designer_max,
designer_min, stesso_brand_num, lift)]
matrice_corr <- cor(variabili_corr, method = "pearson")
```

I risultati sono riportati nella tabella 4.5 seguente, nella quale le meccaniche sono indicate con M+/M- (*mecc_max/mecc_min*), le categorie con C+/C- (*cat_max/cat_min*), i designer con D+/D- (*designer_max/designer_min*) e l'indicatore *stesso_brand* con SB.

	M+	M-	C+	C-	D+	D-	SB	Lift
M+	1,00	0,51	0,53	0,21	0,63	0,22	0,67	0,16
M-	0,51	1,00	0,09	0,41	0,22	0,53	0,25	0,03
C+	0,53	0,09	1,00	0,34	0,49	0,10	0,78	0,13
C-	0,21	0,41	0,34	1,00	0,20	0,39	0,21	0,06
D+	0,63	0,22	0,49	0,20	1,00	0,43	0,68	0,35
D-	0,22	0,53	0,10	0,39	0,43	1,00	0,23	0,30
SB	0,67	0,25	0,78	0,21	0,68	0,23	1,00	0,24
Lift	0,16	0,03	0,13	0,06	0,35	0,30	0,24	1,00

Tab. 4.5 – Matrice di correlazione di Pearson tra le feature di similarità, l'indicatore *stesso_brand* e il lift. Tabella prodotta sulla base dei dati ottenuti su R.

La coppia di variabili più correlata è *cat_max* e *stesso_brand* (0,78). Trattandosi di una correlazione positiva con una variabile binaria (*stesso_brand*: 0 = prodotti indipendenti, 1 = stesso brand/edizione), il segno indica che *cat_max* tende a essere più alto quando *stesso_brand* = 1 – coerentemente con l’attesa che un’espansione o una variante condivida quasi sempre la categoria tematica del gioco base. Invece, *designer_max* risulta la variabile più correlata con il lift (0,35).

Nessun *predittore* è stato escluso sulla base di queste correlazioni: a differenza di un modello lineare, un albero di regressione seleziona un solo *predittore* alla volta a ciascuno split e non risente della stessa instabilità legata a *predittori* correlati tra loro. La matrice resta, però, comunque utile in fase di interpretazione: se l’albero selezionasse *cat_max* come split rilevante, parte di quel segnale potrebbe riflettere informazione già contenuta in *stesso_brand*, e viceversa.

4.7 – Costruzione dell’Albero di Regressione

L’albero è stato costruito secondo la metodologia CART descritta al par. 3.2.4, con un valore di *cp* deliberatamente basso (0,001)²¹ per esplorare tutte le potature sensate, seguito da potatura secondo il criterio 1-SE applicato all’errore di cross-validazione per la selezione finale. Come già anticipato, il target dell’albero è il lift mentre i *predittori* sono le sei *feature* di similarità di Jaccard e l’indicatore *stesso_brand*.

Il valore di *minbucket*, ovvero del numero minimo di osservazioni richieste in ciascuna foglia terminale dell’albero, è stato scelto verificando preliminarmente la stabilità del risultato su un intervallo ampio: (10-35)²². I risultati ottenuti sono riportati di seguito nella tabella 4.6.

²¹ rpart fissa il valore per *cp* a 0,01: abbassarlo a 0,001 evita di scartare split potenzialmente utili già durante la fase di crescita.

²² Il codice completo per la verifica è riportato in Appendice, cod. A.4.9

<i>minbucket</i>	Radice	n split	Import. <i>mecc_max</i>	Import. <i>designer_max</i>
10	<i>mecc_max</i>	3	47,5%	2,1%
11-20	<i>mecc_max</i>	2	51,9%	2,3%
21-35	<i>designer_max</i>	2	35,1%	28,1%

Tab. 4.6 – Stabilità dell'albero al variare di *minbucket* (10-35): radice, numero di split e importanza dei due predittori in competizione. Tabella prodotta sulla base dei risultati ottenuti su R.

L'albero risulta stabile all'interno di due regimi, con un punto di rottura netto a 21, dove la variabile scelta come radice cambia da *mecc_max* a *designer_max* – riflesso della correlazione tra le due variabili (0,63, Tab. 4.5), che competono per lo stesso split. L'errore di cross-validazione alla potatura 1-SE (*xerror*, par. 3.2.4) è quasi identico nei due casi – 0,708 per *minbucket*=15 contro 0,759 per il regime 21-35. La differenza è così piccola da poter essere dovuta al solo rumore campionario, non a una reale differenza nella capacità predittiva delle due strutture: i due alberi, in altre parole, prevedono il lift con un'accuratezza sostanzialmente equivalente.

Si è scelto di adottare *minbucket*=15 per la costruzione dell'albero poiché è tra i valori con il minor errore di cross-validazione e abbastanza lontano dal punto di instabilità (21).

Codice R: costruzione e potatura dell'albero

```
set.seed(123)
albero_full <- rpart(
  lift ~ mecc_max + mecc_min + cat_max + cat_min + designer_max + designer_min +
  stesso_brand,
  data = regole_albero, method = "anova",
  control = rpart.control(minbucket = 15, cp = 0.001)
)

tabella_cp <- albero_full$cptable
xerror_min <- min(tabella_cp[, "xerror"])
xstd_min <- tabella_cp[which.min(tabella_cp[, "xerror"]), "xstd"]
riga_1se <- min(which(tabella_cp[, "xerror"] <= xerror_min + xstd_min))
albero_potato <- prune(albero_full, cp = tabella_cp[riga_1se, "CP"])
```

Risultato: l'albero potato secondo il criterio 1-SE ha 2 split e 3 foglie terminali, vedasi la figura 4.1.

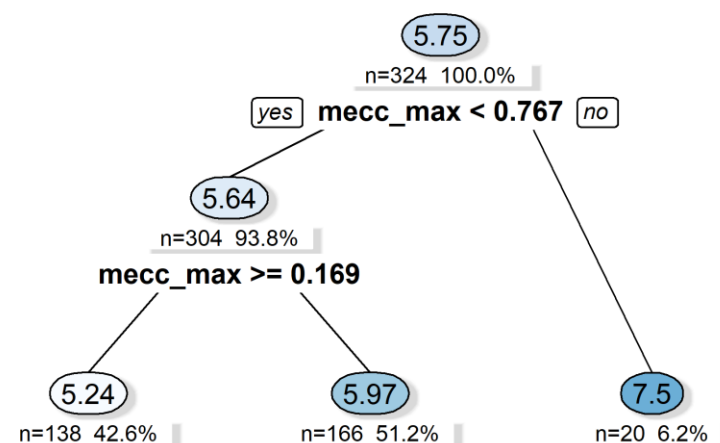


Fig. 4.1 – Rappresentazione grafica dell'albero di regressione (minbucket=15, cp=0,001).
Codice per la realizzazione su R in Appendice, cod. A.4.10.

Prima di procedere all'interpretazione dell'albero, si è voluto verificare che le differenze di lift osservate tra le 3 foglie riflettano un'affinità di co-apprezzamento reale, e non semplicemente il fatto che alcuni dei giochi coinvolti siano poco recensiti: il lift, infatti, tende ad essere più alto quando uno dei due prodotti è più raro, a prescindere da quanto i due siano davvero associati nel gusto degli utenti. Per questo motivo è stato calcolato un secondo indicatore, il lift standardizzato, che corregge questo effetto.

Il calcolo viene eseguito su ciascuna delle 324 regole individualmente, esattamente come già avviene per il lift originale, e poiché ogni regola appartiene già a una delle tre foglie individuate dall'albero, è possibile riportare anche questo nuovo indicatore all'interno delle stesse foglie e calcolarne la media, senza dover ricostruire nulla²³. I valori a confronto vengono riportati nella tabella 4.7.

²³ Il codice completo per il calcolo del lift standardizzato e della media per foglia è riportato in Appendice, cod. A.4.11

Foglia	n	Lift medio	Lift std. medio
Foglia 1	138	5,24	0,111
Foglia 2	166	5,97	0,113
Foglia 3	20	7,50	0,518

Tab. 4.7 – Lift medio e lift standardizzato medio per foglia. Tabella prodotta sulla base dei risultati ottenuti su R.

Confrontando foglia per foglia il lift originale con questo secondo indicatore, la Foglia 3 resta nettamente la più alta su entrambe le misure – anzi il suo vantaggio si amplifica passando al lift standardizzato. Le Foglie 1 e 2, pur diverse tra loro sul lift grezzo (5,24 contro 5,97), mostrano invece un lift standardizzato praticamente identico (0,111 e 0,113): nessuna delle due mostra segnali di lift gonfiato dalla rarità dei giochi coinvolti – entrambe possono quindi essere trattate come pattern genuini nell’interpretazione seguente, senza necessità di escluderne alcuna.

4.8 – Interpretazione dell’Albero e Individuazione dei Pattern

Osservando i nodi che portano a ciascuna foglia e le caratteristiche dei giochi coinvolti nelle regole che le compongono, emergono tre possibili pattern distinti.

4.8.1 – Componente Strutturale

Il primo pattern individuato riguarda la Foglia 3 (Fig. 4.1), caratterizzata da $mecc_max \geq 0,767$: si tratta quindi di giochi che condividono quasi tutte le meccaniche tra loro. A conferma di una somiglianza così marcata, anche l’indicatore $designer_max$ risulta pari a 1 per la totalità delle regole, facendo sì che non solo le meccaniche coincidano quasi del tutto, ma anche che i designer coinvolti siano gli stessi tra antecedente e conseguente.

Coerente con questa lettura è anche l’importanza dell’indicatore $stesso_brand$ nell’albero (8,0%), pur non comparando mai come split effettivo. Si tratta dunque di regole che collegano giochi appartenenti allo stesso brand – espansioni o varianti, come ad esempio: *Dominion: Intrigue*→*Dominion*, *Pandemic Legacy: Season 2*→*Pandemic Legacy: Season 1*, o *Codenames: Pictures*→*Codenames* – e per questo simili tra loro anche nella meccanica. Per questo motivo il pattern viene qui

definito componente strutturale: giochi che condividono, appunto, una struttura comune, sia essa il brand o la meccanica²⁴.

4.8.2 – Affinità di Meccanica

Il secondo pattern individuato riguarda la Foglia 1 (Fig. 4.1), caratterizzata da un livello moderato di somiglianza di meccanica tra antecedente e conseguente ($mecc_max \geq 0,169$). Un controllo²⁵ del pattern, che non coinvolge solo le meccaniche ma anche le categorie più frequenti è riportato nella tabella 4.8 di seguito.

Tipo	Elemento	Occorrenze (su 53 giochi distinti)
Categoria	<i>Economic</i>	29
Meccanica	<i>Hand Management</i>	25
Meccanica	<i>Variable Set-up</i>	23
Meccanica	<i>End Game Bonuses</i>	20
Meccanica	<i>Variable Player Powers</i>	20
Categoria	<i>Card Game</i>	10

Tab. 4.8 – Categorie e meccaniche più ricorrenti tra i giochi della foglia “affinità delle meccaniche”. Tabella prodotta sulla base dei risultati ottenuti su R.

Per valutare che tipo di giochi compongano questa foglia rispetto al resto del catalogo, il profilo dei giochi coinvolti è stato confrontato con quello di un gruppo di riferimento, costituito dai giochi rimanenti dopo l'applicazione del filtro $rating \geq 8$, con l'esclusione dei 68 che compongono le 324 regole. Il confronto è stato condotto tramite il test di Wilcoxon per campioni indipendenti (Wilcoxon, 1945). Il test²⁶ permette di confrontare due gruppi rispetto a una variabile numerica senza assumere che questa segua una distribuzione normale: unite le osservazioni dei due gruppi in un'unica graduatoria ordinata, verifica se i ranghi di un gruppo tendano sistematicamente a collocarsi più in alto o più in basso di quelli dell'altro.

²⁴ Il codice per la verifica del valore di *designer_max* e quello per misurare l'importanza dell'indicatore *stesso_brand* sono riportati in Appendice, cod. A.4.12.

²⁵ Il codice per effettuare il controllo sulle meccaniche e categorie è riportato in Appendice, cod. A.4.13

²⁶ Il codice del test di Wilcoxon applicato alla foglia 1 è riportato in Appendice, cod. A.4.14

I risultati mostrano una complessità dichiarata nettamente superiore per i giochi della foglia (mediana 3,27 contro 1,92 del gruppo di riferimento), una durata di gioco più lunga (120 contro 45 minuti) e un'età minima consigliata più alta (12 contro 10). Il test è stato eseguito escludendo, sia dai giochi della foglia sia dal gruppo di riferimento, le osservazioni con assenza di informazione sui campi numerici coinvolti, dovuta alla presenza dei valori segnaposto individuati nel par. 4.2.5 (0 casi lato foglia su tutti e tre i campi; lato gruppo di riferimento, 65 per *averageweight*, 508 per *playingtime*, 1.027 per *minage*).

4.8.3 – Affinità Residua

Il terzo pattern individuato riguarda la Foglia 2 (Fig. 4.1), caratterizzata da una somiglianza di meccanica quasi assente tra antecedente e conseguente (*mecc_max*<0,169). A differenza di quanto ci si potrebbe attendere, questa caduta della somiglianza della meccanica non si accompagna a un'analogica caduta della somiglianza di categoria: *cat_max* mantiene una mediana di 0,25 e una media di 0,30, sostanzialmente allo stesso livello osservato nella foglia precedente (mediana 0,20, media 0,28). Anche quando l'affinità meccanica viene meno, resta dunque una somiglianza residua di categoria – non abbastanza forte da generare un proprio split nell'albero, ma comunque presente, e possibile concausa di un lift medio (5,97) che risulta il più alto tra le foglie non strutturali. Un controllo²⁷ sulle meccaniche e categorie più frequenti è riportato nella tabella 4.9 sottostante.

Tipo	Elemento	Occorrenze (su 47 giochi distinti)
Categoria	<i>Economic</i>	27
Meccanica	<i>Hand Management</i>	22
Meccanica	<i>End Game Bonuses</i>	16
Meccanica	<i>Open Drafting</i>	16
Meccanica	<i>Variable Set-up</i>	16
Categoria	<i>Card Game</i>	9

Tab. 4.9 – Categorie e meccaniche più ricorrenti tra i giochi della foglia “affinità residua”. Tabella prodotta sulla base dei risultati ottenuti su R.

²⁷ Il codice per effettuare il controllo sulle meccaniche e categorie è riportato in Appendice, cod. A.4.15

Per valutare che tipo di giochi compongano questa foglia rispetto al resto del catalogo, il profilo dei giochi coinvolti è stato confrontato con lo stesso gruppo di riferimento già utilizzato per la foglia precedente tramite il test di Wilcoxon²⁸.

I risultati del test mostrano una complessità dichiarata (*averageweight*) nettamente superiore per i giochi della foglia (mediana 3,25 contro 1,92 del gruppo di riferimento), una durata di gioco più lunga (100 contro 45 minuti) e un'età minima consigliata più alta (12 contro 10).

4.9 – Interpretazione dei Risultati in Ottica di Marketing

La logica di fondo – sfruttare pattern di associazione comportamentale osservati per orientare decisioni di cross-selling – è una pratica consolidata nel retail, e non vi è ragione a priori per ritenere che il settore dei giochi da tavolo si comporti diversamente. Il valore economico potenziale di queste indicazioni non è però quantificabile con precisione sui soli dati di questo lavoro: l'entità specifica del beneficio andrebbe stabilita empiricamente, non stimata per analogia. Il costo di implementazione resta comunque contenuto rispetto alla scelta manuale degli abbinamenti da proporre.

Tra i tre pattern individuati emergono due letture di marketing distinte: quella della componente strutturale, e quella condivisa dalle due foglie di affinità (di meccanica e residua).

4.9.1 – Componente Strutturale

La componente strutturale rappresenta una forma di cross-selling legittima pur trattandosi di un'informazione facilmente intuibile – che un gioco base e la sua espansione o variante siano apprezzati insieme non sorprende – resta comunque una conferma rilevante, che si presta a essere implementata con una raccomandazione a bassissimo rischio e altissima confidenza. Proprio perché

²⁸ Il codice del test di Wilcoxon applicato alla foglia 2 è riportato in Appendice, cod. A.4.16

fondata su una relazione editoriale dichiarata, piuttosto che su una somiglianza osservata empiricamente, è ragionevole ritenerla generalizzabile anche oltre le 324 regole qui analizzate.

4.9.2 – Affinità di Meccanica e Residua

Le due foglie non strutturali, pur restando pattern analiticamente distinti – l'una fondata su una reale somiglianza di meccanica tra i giochi coinvolti, l'altra su una somiglianza di meccanica pressoché assente – condividono un profilo di giochi molto simile: complessità elevata (mediana 3,25-3,27 contro 1,92 del gruppo di riferimento), età minima consigliata più alta (12 contro 10) e la stessa categoria dominante tra i titoli coinvolti (Economic). Data questa convergenza sul piano del profilo, pur restando distinte sul piano della meccanica – il criterio con cui l'albero le separa – le due vengono qui trattate insieme dal punto di vista delle applicazioni di marketing, che risultano sostanzialmente identiche per entrambe. Insieme, le due foglie coprono 304 delle 324 regole – la larghissima parte delle associazioni individuate.

Riguardando la maggior parte delle regole, è ragionevole estendere questa lettura anche ai titoli affini per genere e meccaniche non direttamente coinvolti nelle 324 regole osservate. Resta invece da dimostrare se il principio di fondo – che una somiglianza di meccanica o categoria comporti un analogo comportamento di co-apprezzamento – valga anche al di fuori dei giochi strategici di gestione delle risorse qui osservati.

Queste indicazioni si prestano ad applicazioni diverse a seconda che ci si trovi sul canale online o nel punto vendita fisico.

4.9.3 – Canale e-commerce

Nel canale e-commerce l'implementazione più diretta consiste nella proposta di un titolo complementare nei punti di interazione dell'utente con il sito: nel catalogo prodotti, affiancando i prodotti correlati ad un specifico pattern; nella scheda di un prodotto, tramite la presenza di articoli correlati accanto al titolo visualizzato; nel

carrello, tramite un banner o pop-up con anche la possibilità di offrire un piccolo sconto condizionato all'acquisto congiunto – in tutti e tre i casi proponendo titoli coerenti con il pattern pertinente al prodotto in questione. Si possono inoltre escogitare bundle veri e propri, offrendo direttamente i due giochi come un unico prodotto a prezzo scontato.

Un'applicazione ulteriore che questo canale può offrire riguarda la possibilità di conoscere i gusti o lo storico acquisti degli utenti. Questo permette di individuare i titoli posseduti da ciascun cliente – e se lascia una recensione, anche il relativo grado di apprezzamento – e di proporre, per i titoli coinvolti nei pattern individuati, altri titoli coerenti con il pattern stesso. Ciò avviene tramite comunicazioni mirate a informare sulla disponibilità di novità coerenti con il pattern, consigliare titoli simili, oppure offrire sconti – in modo più rilevante per il cliente rispetto a una promozione generica indirizzata all'intera base utenti.

4.9.4 – Canale Retail Fisico

Nel canale retail fisico, la stessa logica trova applicazione tramite la disposizione a scaffale (shelf layout): collocare vicini il gioco base e la sua espansione o variante, e i titoli strategici con meccaniche o categoria affini, eventualmente accompagnati da segnaletica dedicata – cartellini o espositori che richiamano l'abbinamento all'attenzione del cliente, anche tramite offerte o bundle, sul modello proposto per l'e-commerce. La formazione del personale di vendita è un'opportunità per avvicinarsi al grado di personalizzazione ottenibile con il canale online, e rappresenta la possibilità di indicare al cliente quali abbinamenti proporre in cassa o durante un'eventuale consulenza.

4.9.5 – Implementazione Prudente

Data la natura esplorativa di queste indicazioni – fondate su dati storici di recensione, non su un esperimento controllato sul comportamento di acquisto – si suggerisce una prima implementazione prudente, che introduca inizialmente le azioni proposte in forma contenuta: un banner o uno sconto limitato a un numero

ristretto di coppie di titoli, una campagna e-mail rivolta a un segmento ridotto di clientela, un riposizionamento a scaffale circoscritto a pochi punti vendita. Questa fase pilota consentirebbe di valutare l'efficacia dei suggerimenti attraverso indicatori concreti – il tasso di conversione sul banner o sulla comunicazione proposta, il valore medio del carrello, la quota di ordini contenenti più di un titolo dello stesso pattern – prima di estendere l'iniziativa sulla base dei segnali raccolti.

4.10 – Limiti dell'Analisi

4.10.1 – Scelta del Dataset e della Soglia Rating

Alcune scelte alla base dell'analisi sono state guidate da esigenze pratiche più che da un criterio univocamente corretto. Il dataset recensioni utilizzato non è il più recente tra quelli disponibili su Kaggle: la versione da 26 milioni copre circa 10.000 titoli unici in più (par. 2.2), ma è stata scartata per limiti computazionali riscontrati nella pulizia dei dati già con il volume più contenuto. Allo stesso modo, la soglia $\text{rating} \geq 8$ utilizzata per costruire i panieri (par. 4.3.2) è stata scelta per ottenere un numero maneggevole di regole di buona qualità, non perché fosse l'unica soglia possibile: soglie diverse, come mostrato nello stesso paragrafo, producono un numero di regole molto differente.

4.10.2 – Recensioni come Proxy di Acquisto

Le regole individuate si fondano su recensioni, non su dati di acquisto: un utente può aver recensito un gioco senza mai averlo acquistato. Le associazioni di co-apprezzamento riflettono quindi un giudizio espresso dagli utenti, non necessariamente un comportamento di acquisto congiunto.

4.10.3 – Concentrazione su Pochi Titoli

Infine, le 324 regole coinvolgono complessivamente solo 68 giochi distinti (par. 4.5): i pattern individuati riflettono quindi il comportamento di un nucleo di titoli molto recensiti, e non è detto che si estendano al resto del catalogo.

Capitolo 5

Analisi Predittiva delle Caratteristiche Associate al Successo dei Giochi da Tavolo

Il presente capitolo sviluppa il secondo obiettivo del lavoro, enunciato nel capitolo 1 (par. 1.5): individuare quali caratteristiche di un gioco da tavolo risultino associate al suo successo. A differenza del capitolo precedente, l'obiettivo richiede qui un'impostazione genuinamente predittiva: il dataset viene diviso in due parti su base temporale – un insieme di stima (training set), formato dai titoli più vecchi, su cui i modelli vengono costruiti, e un insieme di verifica (test set), formato dai titoli più recenti e mai osservato durante la stima, su cui la loro capacità di generalizzare viene effettivamente misurata. Il capitolo ripercorre l'intero percorso che ha condotto ai risultati finali: dalla costruzione dei *predittori* a partire dalle caratteristiche dirette dei singoli giochi, fino alla stima e all'interpretazione dell'albero di regressione e della random forest costruiti sul *Geek Rating* come variabile target.

5.1 – Costruzione del Dataset e Trattamento dei Valori Mancanti

A differenza del capitolo 4, l'analisi predittiva utilizzerà le sole informazioni contenute nel dataset giochi – tralasciando quindi il dataset recensioni.

Al dataset giochi di partenza (`games_detailed_info2025.csv`), contenente 27.780 titoli, sono stati mantenuti i soli campi rilevanti ai fini dell'analisi, corretti gli errori di decodifica dei campi testuali e applicato il filtro sul campo anno di pubblicazione, coerentemente con quanto fatto nei primi paragrafi del capitolo precedente.

Codice R: caricamento e selezione dei campi, con decodifica e applicazione filtro anno di pubblicazione

```
giochi <- fread("games_detailed_info2025.csv", encoding = "UTF-8")
giochi <- giochi[, .(id, name, yearpublished, minplayers, maxplayers, playingtime,
minage, boardgamecategory, boardgamemechanic, boardgamedesigner,
usersrated, bayesaverage)]
giochi <- giochi[yearpublished >= 1950]
```

Risultato: il dataset giochi contiene 12 campi, 27.303 osservazioni e anno di pubblicazione compreso tra il 1950 e il 2026.

Tra i campi non mantenuti figura *averageweight*, la misura di complessità già utilizzata nel capitolo 4 nel confronto di profilo tra i pattern individuati (par. 4.8). Il campo viene calcolato da BGG come media dei voti di complessità attribuiti dagli utenti al gioco dopo averlo provato; pur essendo quindi disponibile nel dataset per ogni osservazione, risulterebbe inutilizzabile come *predittore* di successo per un titolo non ancora pubblicato, per il quale un simile dato non esiste ancora.

Poiché nei paragrafi successivi verranno costruiti, a partire dai campi selezionati, i *predittori* da utilizzare nell'albero di regressione e nella random forest, si verifica ora – come già fatto al par. 4.2.5 per il capitolo precedente – la presenza di valori mancanti o segnaposto, sia nei campi numerici sia in quelli testuali. I risultati sono riportati nella tabella 5.1 seguente²⁹.

Campo	Condizione	N. osservazioni
Minage	= 0	1.493
Playingtime	= 0	1.105
Minplayers	= 0	43
Maxplayers	= 0	156
Maxplayers	≥ 99	317
Boardgamemechanic	Lista vuota	1.703
Boardgamecategory	Lista vuota	364
Boardgamedesigner	Nessun designer accreditato (lista vuota o segnaposto “(Uncredited)”/“(Unknown)”)	2.303

Tab. 5.1 – Numero di osservazioni con condizione di informazione mancante o segnaposto. Tabella prodotta sulla base dei risultati ottenuti su R.

Nessuno dei campi coinvolti presenta celle realmente vuote: l'intera assenza di informazione è codificata attraverso i valori segnaposto elencati in tabella.

Per *minage*, *playingtime*, *minplayers* e *maxplayers*, il valore 0 corrisponde a un dato semplicemente non specificato: tra i titoli con *minage* = 0 figurano giochi come

²⁹ Il codice utilizzato per l'estrazione dei valori è riportato in Appendice, cod. A.5.1.

Washington's War e *Fire in the Lake* – *wargame* complessi certamente non pensati per un pubblico di età non specificata – a conferma che si tratta di dati mancanti, non di una indicazione reale. In tutti questi casi il valore viene convertito in *NA*, in modo da segnalare esplicitamente l'assenza di informazione invece di lasciare un valore numerico che potrebbe essere scambiato per un dato realmente osservato. Il trattamento complessivo di questi valori mancanti è discusso al par. 5.2.3, una volta completata la costruzione di tutti i *predittori*.

Il caso *maxplayers* ≥ 99 è invece diverso e richiede un trattamento distinto, poiché qui il valore non segnala un'assenza di informazione, ma comunica un'informazione reale – l'assenza di un limite massimo di giocatori dichiarato dall'editore – che tuttavia non è esprimibile come una quantità numerica. Trattato come tale, un valore come 99 verrebbe interpretato al pari di un conteggio letterale, distorcendo qualunque confronto quantitativo che coinvolga *maxplayers*. Per questo l'informazione viene scorporata in un indicatore binario dedicato³⁰ (*no_max_limit*), che la preserva nella forma corretta, mentre il valore numerico di *maxplayers* per queste sole osservazioni viene impostato a *NA* – non perché sconosciuto, ma perché per un gioco senza limite dichiarato non esiste, in senso stretto, un numero da riportare.

Codice R: trattamento dei valori segnaposto nei campi numerici

```
giochi[minage == 0, minage := NA]
giochi[playingtime == 0, playingtime := NA]
giochi[minplayers == 0, minplayers := NA]
giochi[, no_max_limit := as.integer(maxplayers >= 99)]
giochi[maxplayers == 0, no_max_limit := NA]
giochi[maxplayers == 0, maxplayers := NA]
giochi[maxplayers >= 99, maxplayers := NA]
```

³⁰ L'indicatore avrà valore 1 quando *maxplayers* sarà ≥ 99 , mentre assumerà valore 0 in caso contrario.

Il trattamento delle liste vuote di *boardgamemechanic* e *boardgamecategory*, così come dei segnaposto del campo *boardgamedesigner*, è invece rimandato al par. 5.2 seguente, dove i tre campi vengono trasformati nei rispettivi *predittori*.

5.2 – Costruzione dei Predittori

5.2.1 – Meccaniche e Categorie

I campi *boardgamemechanic* e *boardgamecategory* sono memorizzati da BGG come liste testuali – un gioco può presentare più meccaniche e più categorie contemporaneamente – ed entrambi sono stati estratti con la stessa funzione già definita precedentemente in Appendice, cod. A.4.2.

Ciascun valore mantenuto viene trasformato in una variabile indicatore (*dummy*) binaria: un albero di regressione richiede *predittori* numerici su cui operare gli split, e “possiede questa meccanica” non lo è di per sé – la variabile *dummy* traduce l’informazione in 1 se il gioco la possiede, 0 se non la possiede, rendendola utilizzabile come qualunque altro *predittore*.

Il dataset presenta 192 meccaniche e 84 categorie distinte. Entrambe vengono incluse per intero, senza applicare una soglia di frequenza, lasciando che sia l’albero a scegliere quali risultino effettivamente utilizzabili. La scelta richiede comunque una precisazione: la distribuzione delle frequenze delle meccaniche è fortemente asimmetrica, e le meno comuni compaiono in una manciata di giochi (le tre meno frequenti sono legate a soli 2, 3 e 4 titoli). Un *dummy* quasi sempre pari a 0 esporrebbe al rischio concreto di split spuri³¹, dove una differenza di *Geek Rating* (il target dell’albero) osservata su un pugno di giochi appare ampia per puro effetto della bassa numerosità campionaria, senza riflettere un pattern reale generalizzabile a titoli futuri. È esattamente questo rischio a motivare il *minbucket* adottato in seguito (par. 5.5): escludere strutturalmente dagli split le meccaniche

³¹ Uno split si dice spurio quando la differenza osservata tra i due gruppi che genera nasce dal caso piuttosto che da una relazione reale e ripetibile tra il predittore e il target.

(o categorie) con troppe poche osservazioni per sostenerli in modo affidabile, senza bisogno di deciderlo qui in anticipo escludendole dal dataset.

I giochi con lista vuota (1.703 per le meccaniche, 364 per le categorie; Tab. 5.1) ricevono *NA* su tutte le colonne corrispondenti, anziché 0: un valore 0 asserirebbe con certezza l'assenza di quella meccanica o categoria, mentre una lista vuota segnala solo che l'informazione non è stata registrata. Il trattamento è coerente con quello già adottato per i campi numerici (par. 5.1) e rimandato al par. 5.2.3, una volta completata la costruzione di tutti i *predittori*³².

5.2.2 – Designer

Il campo designer presenta una cardinalità molto elevata: 12.118 valori distinti su 27.303 giochi, il 60,6% dei quali con un solo titolo accreditato. Trattarlo come meccaniche e categorie, con una *dummy* per ciascun valore, non è quindi praticabile.

La soluzione adottata³³ è osservare non l'identità del designer, ma il suo grado di esperienza: quanti altri giochi aveva già pubblicato, in anni strettamente precedenti, al momento in cui il titolo considerato è uscito – mai nello stesso anno né in anni successivi, per non usare come *predittore* un'informazione che a quella data non esisteva ancora. Ad esempio, un titolo del 2015 firmato da un designer con 10 pubblicazioni precedenti riceve *designer_exp* = 10, se lo stesso designer pubblica di nuovo nel 2018, quel titolo riceverà 11. La variabile *designer_exp* viene mantenuta come continua, e come per *playingtime* o *minage*, è l'albero stesso a individuare l'eventuale punto di taglio più utile a spiegare il target. Per i giochi con più designer accreditati si conserva il valore più alto tra quelli coinvolti, non una media, quindi un gioco firmato da un debuttante (*designer_exp*=0) insieme a un designer con 15 titoli alle spalle riceve *designer_exp*=15 – mediarli abbasserebbe

³² Il codice per il trattamento dei valori segnaposto e per la costruzione delle variabili dummy è riportato in Appendice, cod. A.5.2.

³³ Il codice per il controllo sulla cardinalità del campo designer, la costruzione di *designer_exp* e il successivo controllo di sanità sono riportati in Appendice, cod. A.5.3

artificialmente l'esperienza reale portata al progetto dal collaboratore più navigato.

Un controllo di sanità sull'indicatore *designer_exp* osservando un designer molto prolifico (Reiner Knizia, 390 giochi nel dataset) conferma la correttezza del calcolo: il suo primo titolo (*Desperados*, 1980) riceve *designer_exp*=0, e il valore cresce in modo monotono nei titoli successivi.

I segnaposto “(*Uncredited*)” e “(*Unknown*)”, con cui BGG codifica l'assenza di un autore accreditato, vengono esclusi dal conteggio già in fase di estrazione della lista. Un gioco la cui lista conteneva solo questi segnaposto risulta quindi, dopo la rimozione, indistinguibile da un gioco con lista originariamente vuota. I 2.303 giochi, con lista vuota, ricevono tutti *NA* su *designer_exp*, non 0 poiché indicherebbe con certezza l'esordio assoluto di un designer identificato, mentre qui si tratta di assenza di informazione. Anche questi *NA* sono trattati complessivamente al par. 5.2.3.

Questo indicatore permette, in maniera diversa rispetto a quanto fatto per meccaniche e categorie, di calcolare l'apporto del designer rispetto al successo del titolo – anche se non in termini di chi è il designer ma di quanta esperienza aveva il designer.

5.2.3 – Trattamento dei Valori Mancanti nel Dataset

Una volta costruiti tutti i *predittori* (par. 5.2.1, 5.2.2), ciascuno mantenendo distinti i valori *NA* dai valori realmente noti, resta da decidere come trattarli nella fase di stima vera e propria. La scelta iniziale era quella di rimandare la gestione all'albero, nativamente in grado di instradare le osservazioni con valori mancanti attraverso split surrogati. Questa strada non è però percorribile, perché per un controllo di robustezza l'albero sarà affiancato da una random forest, che non gestisce nativamente i valori mancanti nei *predittori*. Poiché il confronto tra i due modelli richiede che siano stimati sullo stesso identico dataset, la questione va risolta prima della stima, in modo coerente per entrambi.

Si è quindi scelto di limitare il dataset di stima alle sole osservazioni complete su tutti i 282 *predittori* costruiti, piuttosto che provare a stimarne il valore – attraverso medie o mediane. Si ottiene così lo stesso dataset per entrambi i modelli, senza alcun valore stimato: le osservazioni si riducono da 27.303 a 21.449. Le osservazioni escluse coincidono in larga parte con giochi privi di meccaniche o categorie note su BGG.

Codice R: costruzione del dataset di stima completo

```
cols_predittori <- c("minplayers", "maxplayers", "playingtime", "minage",  
"no_max_limit",  
grep("^mecc_", names(giochi), value = TRUE),  
grep("^cat_", names(giochi), value = TRUE),  
"designer_exp")  
giochi <- giochi[complete.cases(giochi[, ..cols_predittori])]
```

5.3 – Suddivisione del Dataset in Train/Test su Base Temporale

La suddivisione tra dati di stima e dati di verifica è effettuata su base temporale: i giochi pubblicati fino al 2020 costituiscono il *training set*, quelli dal 2021 in poi il *test set*. Questa scelta consente di verificare se le relazioni individuate sui titoli più vecchi (dal 1950 ad oggi) si mantengano su titoli successivi, mai osservati in fase di stima.

Il taglio al 2020 bilancia due esigenze. Da un lato mantiene una proporzione vicina all'80/20 tra i due componenti: delle 21.449 osservazioni disponibili dopo l'esclusione dei valori mancanti (par. 5.2.3), 17.260 giochi compongono il *training set* e 4.189 il *test set*. Dall'altro evita di collocare il confine troppo vicino al presente, poiché negli anni più recenti il numero di voti ricevuti per titolo diminuisce in modo netto (mediana 134 nel 2020, contro 87 nel 2024 e appena 34 nel 2026), un segnale che il *Geek Rating* dei titoli usciti da poco tempo non si è ancora consolidato. Non si tratta dell'unico taglio possibile, ma di un punto di equilibrio identificato come ragionevole tra le due esigenze.

Codice R: suddivisione *train/test*

```
train <- giochi[yearpublished <= 2020]
test <- giochi[yearpublished >= 2021]
```

Un controllo sui *predittori* così suddivisi ha evidenziato che, in seguito all'esclusione delle osservazioni con valori mancanti, quattro dei 282 *predittori* – *no_max_limit*, una categoria e due meccaniche – risultano privi di variazione sia sul training set sia sul test set. Un *predittore* costante non fornisce alcuna informazione utilizzabile: non potendo generare uno split nell'albero né contribuire alla random forest, tutti e quattro vengono esclusi dai due set, portando il numero complessivo di *predittori* a 278. L'informazione sul numero massimo di giocatori resta comunque rappresentata da *maxplayers*: *no_max_limit* era stato introdotto per gestire i titoli privi di un limite massimo dichiarato (par. 5.1), una condizione che, in seguito all'esclusione delle osservazioni incomplete, non compare più tra le osservazioni rimaste.

Codice R: controllo e rimozione dei *predittori* costanti su *train* e *test*

```
costanti_train <- sapply(cols_predittori, function(v) length(unique(train[[v]])) == 1)
costanti_test <- sapply(cols_predittori, function(v) length(unique(test[[v]])) == 1)
costanti <- cols_predittori[costanti_train | costanti_test]
giochi[, (costanti) := NULL]
cols_predittori <- setdiff(cols_predittori, costanti)

train <- giochi[yearpublished <= 2020]
test <- giochi[yearpublished >= 2021]

train_model <- train[, c("bayesaverage", cols_predittori), with = FALSE]
test_model <- test[, c("bayesaverage", cols_predittori), with = FALSE]
```

5.4 – Matrice di Correlazione tra i Predittori dell'Albero

Con 278 *predittori*, a differenza degli 8 del par. 4.6, la matrice di correlazione completa non è rappresentabile in una tabella: i risultati vengono descritti qui sinteticamente, calcolati sul *training set* (17.260 osservazioni), l'unico su cui i modelli verranno effettivamente stimati.

La correlazione più alta in valore assoluto è 0,649, tra *Hexagon Grid* (meccanica) e *Wargame* (categoria) – una relazione coerente, dal momento che le mappe esagonali sono una convenzione diffusa nei wargame. Nessuna coppia supera la soglia 0,7: non è quindi necessario escludere alcun *predittore* per ridondanza. Le altre correlazioni più marcate coinvolgono coppie altrettanto interpretabili: *Simulation* e *Wargame* a 0,591; *Spelling* e *Word Game* a 0,556; *minage* e *Children's Game* a -0,479, coerente con il fatto che i giochi per bambini richiedono tipicamente un'età minima più bassa.

Codice R: matrice di correlazione tra i predittori

```
cor_mat <- cor(train[, ..cols_predittori])
```

5.5 – Costruzione e Potatura dell'Albero di Regressione

5.5.1 – La variabile Target: il Geek Rating

La variabile target con cui viene costruito l'albero è *bayesaverage*, definito anche come *Geek Rating* e assegnato da BGG a ciascun gioco. A differenza del voto medio grezzo, il *Geek Rating* è calcolato come media bayesiana: un numero fisso di voti fittizi di valore neutro (pari a circa 5,5) viene aggiunto ai voti reali degli utenti prima di calcolarne la media. Questo meccanismo corregge la tendenza dei giochi con pochi voti ad assumere valori estremi per effetto della sola variabilità campionaria, ed è la ragione per cui il *Geek Rating*, e non il voto medio grezzo, viene qui utilizzato come target.

Un controllo preliminare conferma che il campo non presenta valori mancanti sulle 21.449 osservazioni del dataset di stima originale: *bayesaverage* è definito per ogni gioco, con un intervallo compreso tra 4,34 e 8,41 – interamente compreso nella scala 1-10 attesa, senza valori nulli, negativi o fuori scala.

Codice R: controllo di validità della variabile target

```
sum(is.na(giochi$bayesaverage))  
range(giochi$bayesaverage)
```

5.5.2 – Selezione del Minbucket

Il valore di *minbucket* – il numero minimo di osservazioni richieste in ciascuna foglia terminale – non è stato fissato a priori, ma scelto confrontando su un intervallo di valori la dimensione dell'albero potato, secondo il criterio 1-SE, e l'errore di cross-validazione corrispondente, calcolati esclusivamente sul training set. I valori sono riportati nella tabella 5.2 seguente.

<i>minbucket</i>	n. split	n. foglie	Osservazioni Minime per foglia	Errore cross- validazione
3	32	33	6	0,7454
5	33	34	6	0,7429
10	37	38	10	0,7356
20	27	28	26	0,7532
30	24	25	30	0,7562
40	30	31	45	0,7589
50	21	22	58	0,7708
75	23	24	77	0,7763
100	17	18	175	0,7885

Tab. 5.2 – Dimensione dell'albero potato (criterio 1-SE) ed errore di cross-validazione al variare di *minbucket*, calcolati sul training set con seed fisso (123). Tabella prodotta sulla base dei risultati ottenuti su R.

Valori molto bassi di *minbucket* producono foglie troppo piccole per una stima affidabile: a *minbucket* uguale a 5 e 10, il minimo di osservazioni per foglia scende rispettivamente a 6 e 10 (Tab. 5.2), un campione insufficiente su cui calcolare una media minimamente robusta – indipendentemente dall'apparente vantaggio in termini di errore di cross-validazione, che in questi casi riflette più l'adattamento a variazioni casuali del *training* che un pattern reale. Tra i valori con foglie di dimensione ampiamente sufficiente (20 e oltre), il confronto sull'errore di cross-validazione non individua un candidato nettamente superiore agli altri: le differenze in gioco sono contenute e non garantiscono una scelta univoca. Si è adottato *minbucket*=20 come valore di riferimento per l'albero finale – non l'unica

scelta possibile tra i candidati ragionevoli, ma una che garantisce foglie sufficientemente popolate restando nella fascia di errore più bassa osservata³⁴.

5.5.3 – Costruzione dell’Albero

L’albero è stato costruito secondo la metodologia CART già descritta al par. 3.3.1, sui 278 *predittori* del par. 5.3, con un valore di *cp* deliberatamente basso (0,001) per esplorare tutte le potature sensate.

Con *minbucket*=20, l’albero potato secondo il criterio 1-SE presenta 27 split e 28 foglie terminali, nessuna delle quali sotto le 26 osservazioni³⁵. Data la sua dimensione, non è possibile rappresentarlo per intero in una figura leggibile: la figura 5.1 sottostante ne mostra quindi solo i primi tre livelli, corrispondenti alla parte iniziale dell’albero³⁶.

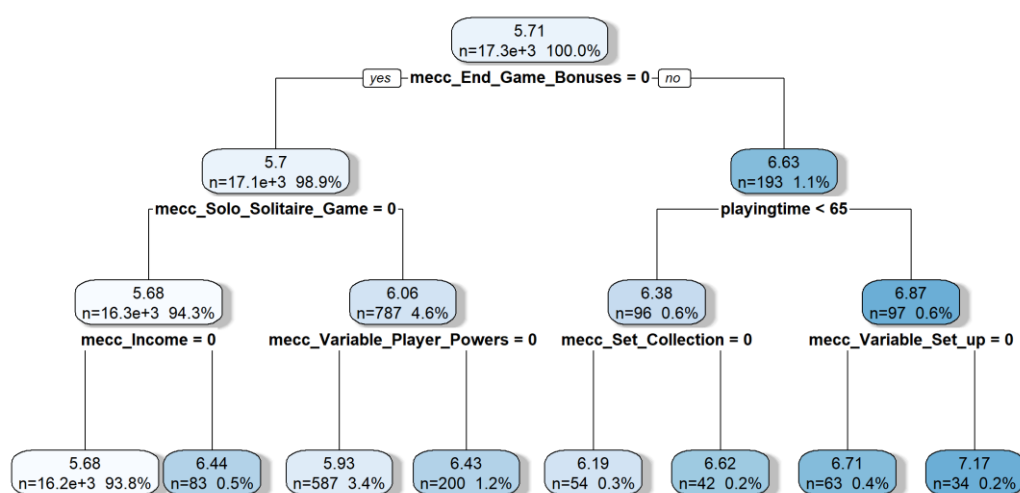


Fig. 5.1 – Rappresentazione dei primi tre livelli dell’albero. Rappresentazione eseguita tramite codice R.

³⁴ Il codice per la scansione dei valori di *minbucket* è riportato in Appendice, cod. A.5.4.

³⁵ Il codice per la costruzione dell’albero è riportato in Appendice, cod. A.5.5.

³⁶ Il codice per la rappresentazione dell’albero è riportato in Appendice, cod. A.5.6

La radice dell'albero è *mecc_End_Game_Bonuses*: i 193 giochi del training con questa meccanica hanno un *Geek Rating* medio di 6,63, contro 5,70 per tutti gli altri³⁷. Un controllo di stabilità temporale, condotto interamente all'interno del *training*, mostra che l'effetto resta presente e nella stessa direzione sia nei titoli più vecchi (differenza di 1,08 punti fino al 2017) sia in quelli più recenti (0,72 punti nel 2018-2020), ridimensionandone solamente l'intensità.

5.5.4 – Valutazione sul Test Set

Per stabilire se l'albero, nel suo complesso, generalizzi bene al di fuori del *training*, il suo errore di previsione sul *test set* viene confrontato con quello di una previsione banale: assegnare a ogni gioco di test lo stesso valore fisso, pari alla media del *Geek Rating* osservata nel training (5,707) – una previsione “cieca”, che non tiene conto di alcun *predittore* e resta identica per ogni titolo. L'albero, al contrario, produce una previsione diversa per ciascun gioco, a seconda di dove questo ricade seguendo gli split appresi in fase di stima.

L'errore di entrambi è misurato con l'*RMSE*: la differenza tra il valore osservato e quello previsto, elevata al quadrato per ciascun gioco, mediata su tutte le osservazioni, e riportata alla scala originale tramite radice quadrata – un numero che indica, in media, di quanto ci si allontana dal valore vero.

L'albero sbaglia in media di più della previsione costante (0,439 contro 0,368) – nonostante sul *training* il suo errore fosse inferiore (0,327)³⁸.

Un esempio concreto aiuta a capire come questo accada. *Noli*, un gioco del *test set* con *Geek Rating* reale pari a 5,522, riceve dalla previsione fissa un valore di 5,707 (un errore di 0,19); l'albero, seguendo i suoi split, lo colloca in una foglia che nel training conteneva titoli con rating tipicamente più alto, e gli assegna una previsione di 6,910 – un errore di 1,39, quasi sette volte più grande. Non è un caso

³⁷ Il codice per il confronto tra chi ha meccanica radice e chi non ce l'ha, con controllo di stabilità temporale è riportato in Appendice, cod. A.5.7.

³⁸ Il codice per il calcolo dei valori dell'*RMSE* e per il successivo esempio sono riportati in Appendice, cod. A.5.8

isolato: episodi come questo, ripetuti sui 4.189 giochi test, fanno sì che l'albero, sommando tutti gli errori, sbagli in media più della previsione fissa.

Seppur pertinente alla previsione del *Geek Rating* piuttosto che al riconoscimento delle variabili associate al successo, l'esempio evidenzia come l'albero tenda ad adattarsi a pattern specifici del training set non replicabili sui titoli più recenti. Tale evidenza giustifica l'introduzione della random forest quale verifica di robustezza dei risultati, propedeutica all'esame delle variabili maggiormente associate al successo.

5.6 – Random Forest come Verifica di Robustezza

5.6.1 – Costruzione della Random Forest

La random forest, descritta nella sua logica generale al par. 3.3.2, corregge l'instabilità di un singolo albero costruendone molti e mediandone le previsioni. A differenza dell'albero del par. 5.5, che usa l'intero *training set* e valuta tutti i 278 *predittori* a ogni *split*, la random forest introduce due fonti di casualità indipendenti: ciascuno dei suoi alberi viene costruito su un campione *bootstrap* – ottenuto pescando 17.260 osservazioni dal *training set* con reinserimento, per cui alcuni giochi compaiono più volte e altri (mediamente circa il 37%) restano invece esclusi dal campione di quello specifico albero – e, a ogni singolo split, viene considerato solo un sottoinsieme casuale dei *predittori* invece dell'intero elenco. La previsione finale per un gioco è la media delle previsioni dei singoli alberi.

Questo doppio meccanismo casuale è la ragione per cui la random forest tende a essere più stabile di un singolo albero: le peculiarità di un campione o di una sequenza di split, che in un albero solo possono dominare l'intera struttura, si mediano su centinaia di alberi costruiti in condizioni leggermente diverse tra loro.

Sono stati stimati 500 alberi (pacchetto ranger), con un numero di *predittori* considerato ogni split pari a 16 – la radice quadrata dei 278 disponibili, valore di default per problemi di regressione. A differenza del *minbucket* per l'albero singolo, la random forest non richiede una scansione dedicata dei parametri: la sua stabilità

deriva dalla struttura del metodo stesso (il grande numero di alberi mediati), non dalla scelta accurata di un singolo parametro.

Codice R: Costruzione della random forest

```
library(ranger)
set.seed(123)
rf <- ranger(bayesaverage ~ ., data = train_model, num.trees = 500, importance =
"impurity", seed = 123)
```

5.6.2 – Valutazione sul Test Set

Viene svolto lo stesso calcolo dell'RMSE sul *train/test* svolto per l'albero singolo. La random forest migliora rispetto all'albero singolo sul test (0,395 contro 0,439), confermando che parte della sua debolezza derivava proprio dall'instabilità di un modello costruito su un solo campione e una sola sequenza di split. Il miglioramento non è però sufficiente a superare la previsione costante (0,368): la random forest si avvicina alla previsione costante, ma non la supera.

Codice R: valutazione della random forest sul test set

```
pred_train_rf <- predict(rf, data = train_model)$predictions
pred_test_rf <- predict(rf, data = test_model)$predictions
rmse_train_rf <- sqrt(mean((train_model$bayesaverage - pred_train_rf)^2))
rmse_test_rf <- sqrt(mean((test_model$bayesaverage - pred_test_rf)^2))
```

5.6.3 – Confronto dell'Importanza delle Variabili

Non essendo la random forest rappresentabile come un singolo albero, l'unica lettura possibile del suo funzionamento è l'importanza delle variabili: quanto la disponibilità di ciascun *predittore*, in media su tutti i 500 alberi, contribuisce a ridurre l'errore di previsione. Viene calcolata per prima con il metodo a impurità, disponibile senza calcoli aggiuntivi come sottoprodotto della stima nella random forest – ma con un limite noto: può sovrastimare l'importanza delle variabili continue rispetto alle dummy binarie, perché offrono loro molti più punti di taglio candidati tra cui l'algoritmo può scegliere (*designer_exp* può essere spezzato a 1, a 2, a 3, mentre una dummy come *mecc_Solo_Solitaire_Game* ha un solo punto di

taglio possibile). Per questo motivo, il risultato a impurità non viene preso come definitivo, ma verificato con il metodo a permutazione – computazionalmente più oneroso poiché richiede di ricalcolare le previsioni rimescolando casualmente i valori di una variabile alla volta e osservando quanto peggiora l'errore, ma non soggetto allo stesso limite.

designer_exp risulta la variabile più importante con entrambi i metodi. È un risultato che merita attenzione proprio perché in apparente contraddizione con quanto trovato molto prima (par. 5.2.2): la sua correlazione con il target era debolissima ($r=0,06$). La spiegazione più plausibile è che il suo potere predittivo emerga in combinazione con altre variabili – un tipo di relazione che un modello ad alberi può cogliere e un semplice coefficiente di correlazione no.

Confrontando le prime dieci variabili per importanza nella random forest con quelle usate dall'albero singolo (par. 5.5.3), sette compaiono in entrambe le classifiche: *mecc_End_Game_Bonuses*, *mecc_Solo_Solitaire_Game*, *mecc_Variable_Player_Powers*, *designer_exp*, *playingtime*, *mecc_Open_Drafting* e *maxplayers*. Questa convergenza tra due modelli costruiti in modo molto diverso – uno singolo e deterministico, l'altro una media di centinaia di alberi su campioni e sottoinsiemi di predittori casuali – è l'evidenza più solida a disposizione per stabilire quali caratteristiche associare al successo con maggiore fiducia, ed è la base da cui parte l'interpretazione dei risultati (par. 5.7).

Codice R: importanza delle variabili delle variabili in random forest e albero singolo

```
imp_impurita <- sort(importance(rf), decreasing = TRUE)
head(imp_impurita,10)
set.seed(123)
rf_perm <- ranger(bayesaverage ~ ., data = train_model, num.trees = 500, importance =
"permutation", seed = 123)
imp_permutazione <- sort(importance(rf_perm), decreasing = TRUE)
head(imp_permutazione,10)
imp_albero <- sort(albero$variable.importance, decreasing = TRUE)
head(imp_albero, 10)
```

5.7 – Stabilità Temporale delle Associazioni Individuate

Il confronto tra albero e random forest (par. 5.6.3) ha individuato sette *predittori* la cui importanza risulta convergente tra i due modelli: *mecc_End_Game_Bonuses*, *mecc_Solo_Solitaire_Game*, *mecc_Variable_Player_Powers*, *designer_exp*, *playingtime*, *mecc_Open_Drafting* e *maxplayers*. Che una caratteristica risulti importante per entrambi i modelli non garantisce però che l'associazione osservata regga nel tempo: lo si è già visto per *End Game Bonuses* (par. 5.5.3), il cui effetto si riduce sensibilmente per i titoli di anni diversi all'interno del *training*. Lo stesso controllo viene ora esteso, sistematicamente, a tutte le variabili osservando le variazioni tra *training* e *test set*.

Il criterio adottato è il seguente: se il segno dell'associazione resta lo stesso tra *training* e *test*, la caratteristica viene considerata un'indicazione valida, distinguendo comunque quanto la sua intensità si sia mantenuta, ridotta o aumentata; un'eventuale inversione di segno segnalerebbe invece un'associazione apparente nel *training* ma non confermata, da sconsigliare esplicitamente.

5.7.1 – Variabili Meccaniche

Per le quattro meccaniche si confronta il *Geek Rating* medio dei giochi che le possiedono con quello dei giochi che non le possiedono, separatamente su *training* e *test set*.

Meccanica	Differenza (training)	Differenza (test)	Variazione
<i>End_Game_Bonuses</i>	0,929	0,215	-77%
<i>Solo_Solitaire_Game</i>	0,422	0,167	-61%
<i>Open_Drafting</i>	0,184	0,142	-23%
<i>Variable_Player_Powers</i>	0,195	0,168	-13%

Tab. 5.3 – Confronto con/senza le quattro meccaniche sul train e sul test set. Tabella prodotta sulla base dei risultati ottenuti su R.

Nessuna delle quattro inverte il segno, ma il grado di intensità è diverso. *Variable Player Powers* e *Open Drafting* mantengono un'associazione sostanzialmente stabile, con una riduzione più contenuta; *End Game Bonuses* e *Solo/Solitaire Game*

– pur restando associate a un *Geek Rating* più alto anche sul test – perdono rispettivamente il 77% e il 61% della propria intensità, un ridimensionamento troppo ampio per essere presentati con la stessa fiducia delle altre due.

Codice R: confronto con/senza per le quattro meccaniche sul *train* e sul *test set*

```
for (v in c("mecc_End_Game_Bonuses", "mecc_Solo_Solitaire_Game",  
"mecc_Variable_Player_Powers", "mecc_Open_Drafting"))  
{  
  diff_train <- mean(train$bayesaverage[train[[v]]==1]) -  
  mean(train$bayesaverage[train[[v]]==0])  
  diff_test <- mean(test$bayesaverage[test[[v]]==1]) -  
  mean(test$bayesaverage[test[[v]]==0]) }
```

5.7.2 – Variabili Strutturali

Per *designer_exp*, *playingtime* e *maxplayers*, un confronto con/senza non è applicabile: si osserva invece il *Geek Rating* medio per fasce di valore, un metodo che permette di cogliere anche eventuali relazioni non lineari.

designer_exp mostra una relazione pressoché monotona e quasi identica tra *training* e *test*, dal debutto assoluto (5,63) fino alla fascia 21-50 (circa 5,81-5,82), con un lieve calo oltre i 50 – una regione dominata da un numero ristretto di designer molto prolifici, e quindi meno generalizzabile.

playingtime segue una forma a U rovesciata, non un andamento lineare crescente o decrescente: il *Geek Rating* aumenta con la durata fino a un intervallo centrale (60-180 minuti), per poi ridiscendere sia verso i giochi molto brevi sia verso quelli molto lunghi. La forma si ripete in modo sostanzialmente identico su *training* e *test*, con il picco che si sposta leggermente (60-90 minuti nel *training*, 120-180 nel *test*) mantenendo però la stessa dinamica di crescita e calo agli estremi.

maxplayers mostra una forma meno netta, con un lieve picco tra i 3 e i 5 giocatori e un calo verso entrambi gli estremi (giochi per 1-2 o per più di 6 giocatori), ripetuta con buona approssimazione in entrambi i periodi.

Codice R: *Geek Rating* medio per fasce su *train* e *test set*.

```
brks_de <- c(-1,0,2,8,20,50,999)
train[, fascia := cut(designer_exp, brks_de)]
train[, .(media = mean(bayesaverage), n = .N), by = fascia]
# stessa logica per playingtime (fasce di durata) e maxplayers (fasce di numero massimo
di giocatori),
# ripetuta su train e su test
```

Nessuna delle sette variabili individuate inverte segno tra *training* e *test set*. Le tre variabili strutturali (*designer_exp*, *playingtime*, *maxplayers*) mostrano relazioni marcatamente stabili, mentre tra le variabili meccaniche solo *Variable Player Powers* e *Open Drafting* reggono bene, le altre, pur non invertendo di segno, si indeboliscono troppo per poter essere presentate con la stessa fiducia. Questa distinzione guiderà la sintesi finale del capitolo (par. 5.8).

5.7.3 – Posizione nella Distribuzione del Geek Rating

Il *Geek Rating* è distribuito in modo fortemente asimmetrico: la metà dei giochi non supera il 5,55, il 90% resta sotto 6,17, e solo l'1% raggiunge 7,3 o più, fino a un massimo di 8,41. In questo contesto, la domanda rilevante non è di quanti punti si sposta la media, ma in che punto della distribuzione colloca un gioco la presenza – o l'assenza – di ciascuna delle sette caratteristiche individuate.

In tutti e sette i casi, su entrambi i periodi, la condizione favorevole colloca il gioco sistematicamente più in alto nella distribuzione – collocandolo, in *training*, tra il 78° e il 95° percentile per la condizione favorevole, contro il 65°-74° per quella sfavorevole; sul test, rispettivamente tra il 77°-84° e il 64°-73° percentile. Lo spostamento va dai circa 10 punti percentile di *maxplayers* ai 21 di *End Game Bonuses* in *training*. Il divario si restringe per *End Game Bonuses* e *Solo/Solitaire Game* passando dal *training* al *test*, entrambe le meccaniche sono diventate più diffuse nel periodo recente, il che ne diluisce parte del valore come segnale distintivo.

Codice R: posizione percentile dei gruppi con/senza ciascuna caratteristica

```
perc <- function(x, dataset) round(100 * mean(dataset$bayesaverage < x), 1)
for (v in c("mecc_End_Game_Bonuses", "mecc_Solo_Solitaire_Game",
            "mecc_Variable_Player_Powers", "mecc_Open_Drafting")) {
  media_con <- mean(train$bayesaverage[train[[v]] == 1])
  media_senza <- mean(train$bayesaverage[train[[v]] == 0])
  perc(media_con, train); perc(media_senza, train)
}
# stessa logica sulle fasce migliore/peggiore di designer_exp, playingtime, maxplayers,
# e ripetuta sul test
```

5.8 – Sintesi del Capitolo, Applicazione e Limiti

5.8.1 – Sintesi dei Risultati

L'albero di regressione (par. 5.5) e la random forest (par. 5.6), stimati sugli stessi 278 *predittori* costruiti a partire dalle caratteristiche dirette dei singoli giochi (par. 5.2), non riescono a prevedere con precisione il *Geek Rating* di titoli non osservati in fase di stima: entrambi restano sotto la previsione costante (RMSE 0,439 e 0,395 rispettivamente, contro 0,368) – la random forest migliora sensibilmente rispetto al singolo albero, ma non abbastanza da superare la previsione più semplice possibile.

Questo risultato, di per sé modesto, non intacca però l'obiettivo del capitolo (par. 1.5): individuare quali caratteristiche siano associate al successo, non prevedere con esattezza il rating di un titolo specifico. Da questo punto di vista, il confronto tra i due modelli (par. 5.6.3) individua sette predittori la cui importanza risulta convergente - *mecc_End_Game_Bonuses*, *mecc_Solo_Solitaire_Game*, *mecc_Variable_Player_Powers*, *designer_exp*, *playingtime*, *mecc_Open_Drafting* e *maxplayers* – e il controllo di stabilità temporale (par. 5.7) conferma che nessuna delle sette inverte segno passando dal *training* al *test set*, collocando sistematicamente i giochi che le possiedono più in alto nella distribuzione del *Geek Rating* (par. 5.7.3).

Il grado di fiducia non è però uniforme. *designer_exp*, *playingtime*, *maxplayers*, *Variable Player Powers* e *Open Drafting* mantengono un'associazione sostanzialmente stabile tra i due periodi. *End Game Bonuses* e *Solo/Solitaire Game*, pur non invertendo segno, si indeboliscono in modo marcato (rispettivamente del 77% e del 61%).

5.8.2 – Applicazione alla Fase di Ideazione del Prodotto

Le sette caratteristiche individuate non hanno tutte la stessa natura per chi progetta un gioco. Quattro sono vere e proprie meccaniche di gioco che uno sviluppatore può decidere di inserire nel titolo: *End Game Bonuses*, *Solo Solitaire Game*, *Variable Player Powers* e *Open Drafting*. Due, invece, riguardano caratteristiche strutturali del titolo: la durata (*playingtime*) e il numero massimo di giocatori (*maxplayers*). Infine, *designer_exp*, che riguarda l'esperienza di chi progetta le regole e le meccaniche del gioco – non un ingrediente che si aggiunge al titolo come una meccanica, ma una scelta che si gioca a monte nella composizione del team di progetto: coinvolgere un designer o un team con una determinata esperienza nel settore.

Per quanto riguarda le caratteristiche strutturali del gioco, *playingtime* e *maxplayers* offrono un'indicazione quantitativa precisa: una durata compresa tra 60 e 180 minuti e un numero di giocatori tra 3 e 5 collocano il gioco nella fascia associata a un *Geek Rating* più alto in modo stabile su entrambi i periodi (par. 5.7.2, 5.7.3); allontanarsi da questi intervalli – verso partite molto brevi o molto lunghe, verso titoli per 1-2 o per più di 6 giocatori – corrisponde a un posizionamento sistematicamente più basso nella distribuzione.

Tra le quattro meccaniche, *Variable Player Powers* e *Open Drafting* possono essere consigliate con una fiducia comparabile a quella delle due variabili precedenti, avendo mantenuto un'associazione stabile tra il training e il test set. *End Game Bonuses* e *Solo/Solitaire Game* restano associate a un *Geek Rating* più alto anche sul test, ma con un'intensità ridimensionata in modo marcato rispetto al training (par. 5.7.1 e 5.7.3).

Per *designer_exp*, l'indicazione utile non è un valore singolo ma una fascia di valori: l'associazione con un *Geek Rating* più alto è più solida per designer o team con un numero di titoli precedenti pubblicati compreso tra 9 e 50 (par. 5.7.2); il debutto assoluto (*designer_exp* = 0) rappresenta il punto più debole della distribuzione, mentre oltre il 50esimo titolo il segnale diventa meno affidabile, dominato da un numero ristretto di autori estremamente prolifici (par. 5.5.3) e quindi meno generalizzabile come indicazione di carattere ampio.

5.8.3 – Limiti Riscontrati

I risultati del capitolo vanno letti alla luce di alcuni limiti, in parte già emersi nel corso dell'analisi.

Il dataset riflette esclusivamente la community di BoardGameGeek – non un campione rappresentativo dell'intero mercato dei giochi da tavolo. Il *Geek Rating* misura quindi il gradimento di questa specifica community, non necessariamente il successo commerciale di un titolo né l'apprezzamento del pubblico più ampio: un gioco può vendere bene senza raccogliere molte valutazioni su BGG, o viceversa.

Il filtro di completezza applicato al dataset di stima (par. 5.2.3) esclude sistematicamente i giochi con meccaniche, categorie o altre informazioni non documentate su BGG – non una porzione casuale, ma una porzione di dataset che tende a coincidere con titoli meno recenti. Le associazioni individuate valgono quindi per la porzione di catalogo effettivamente osservabile, non necessariamente per l'insieme completo dei giochi pubblicati.

Va inoltre ricordato che il *Geek Rating* di un gioco è un valore unico, esito congiunto di tutte le caratteristiche insieme: i confronti del par. 5.7 isolano una variabile alla volta, senza poter scomporre quanto ciascuna abbia effettivamente contribuito al voto di un singolo gioco.

Infine, le sette caratteristiche discusse al par. 5.8.2 sono associazioni statistiche, verificate per stabilità nel tempo (par. 5.7) ma non relazioni causali dimostrate: che un gioco con *Variable Player Powers* tenda ad avere un *Geek Rating* più alto non

implica necessariamente che aggiungere quella meccanica a un titolo qualunque ne migliori la valutazione – questo aspetto andrebbe testato sottoponendo lo stesso titolo con e senza la meccanica ad un gruppo di giocatori.

Capitolo 6

Conclusioni

Il presente capitolo chiude il lavoro riprendendo, alla luce dei risultati ottenuti nei capitoli 4 e 5, i due obiettivi enunciati nel capitolo 1: individuare pattern di associazione utilizzabili in ottica di cross-selling (par. 1.4) e le caratteristiche di un gioco da tavolo associate al suo successo (par. 1.5). Dopo un richiamo sintetico dei risultati (par. 6.1) e una riflessione sul rapporto tra le due analisi (par. 6.2), vengono discussi i limiti trasversali dell'intero lavoro (par. 6.3) e alcuni possibili sviluppi futuri (par. 6.4).

6.1 – Richiamo ai Risultati

L'analisi di cross-selling condotta nel capitolo 4 ha individuato, tramite l'algoritmo Apriori applicato ai panieri di co-apprezzamento costruiti sulle recensioni con voto ≥ 8 , 324 regole di associazione con lift compreso tra 4,05 e 10,55 (par. 4.3.2). L'albero di regressione costruito sulle feature di similarità tra i giochi coinvolti (par. 4.7) ha permesso di ricondurre queste regole a tre pattern distinti (par. 4.8): una *componente strutturale* fondata su una relazione editoriale dichiarata tra i titoli – espansioni o varianti; e due *pattern di affinità*: di *meccanica* e *residua*, che pur restando distinti sul piano della somiglianza delle meccaniche condividono un profilo di giochi molto simile – complessità, durata ed età minima consigliata nettamente superiori al gruppo di riferimento, motivo per cui vengono trattati insieme nelle applicazioni di marketing (par. 4.9).

L'analisi predittiva del capitolo 5 non è riuscita a prevedere con precisione il *Geek Rating* di titoli non osservati in fase di stima – sia l'albero di regressione sia la random forest restano sotto la previsione costante – pur non intaccando questo l'obiettivo del capitolo. Il confronto tra i due modelli ha permesso di individuare sette *predittori* la cui importanza risulta convergente e stabile nel tempo (par. 5.6.3, 5.7): quattro meccaniche di gioco (*End Game Bonuses*, *Solo/Solitaire Game*, *Variable Player Powers*, *Open Drafting*), due caratteristiche strutturali (*playingtime*,

maxplayers) e l'esperienza del designer (*designer_exp*). Applicate alla fase di ideazione del prodotto (par. 5.8.2), *playingtime* e *maxplayers* offrono un'indicazione quantitativa precisa (una durata tra 60 e 180 minuti, un numero di giocatori tra 3 e 5), mentre il designer riguarda la scelta a monte del team, indicando come preferibili individui con 9-50 titoli pubblicati precedentemente.

6.2 – Complementarietà delle Due Analisi

Le due analisi rispondono a momenti diversi della vita di un gioco da tavolo: il capitolo 4 guarda a un titolo già pubblicato e recensito, per orientare decisioni di vendita associata; il capitolo 5 guarda a un titolo non ancora pubblicato, per orientare le decisioni prese in fase di progettazione. Sul piano applicativo, le due analisi restano indipendenti tra loro, ma si prestano a un utilizzo complementare: il capitolo 5 può orientare le decisioni prese a monte, in fase di progettazione di un nuovo titolo mentre il capitolo 4 quelle prese a valle, una volta che il titolo è stato pubblicato.

6.3 – Limiti Trasversali

Entrambe le analisi si fondano sui dataset derivanti da BoardGameGeek: i panieri di co-apprezzamento del capitolo 4 e il *Geek Rating* utilizzato come target nel capitolo 5 riflettono esclusivamente il giudizio della community BGG, non un campione rappresentativo dell'intero mercato dei giochi da tavolo. Un titolo può quindi risultare associato ad altri giochi, o presentare caratteristiche legate a un *Geek Rating* elevato, senza che questo rifletta necessariamente il suo apprezzamento presso il pubblico più ampio o il suo successo commerciale.

In secondo luogo, come già anticipato al par. 1.3, la community BGG non è distribuita in modo uniforme nel mondo, ma è composta in larga parte da utenti nordamericani ed europei. I pattern di cross-selling e le caratteristiche di successo individuati in questo lavoro vanno quindi considerati propri di questi specifici mercati, non necessariamente generalizzabili a contesti culturali e geografici

differenti, dove abitudini di consumo e criteri di apprezzamento potrebbero non coincidere con quelli qui osservati.

Infine, nessuna delle due analisi tiene conto del ruolo dell'editore – del budget destinato al marketing, della qualità della distribuzione e della visibilità commerciale ottenuta da un titolo. Si tratta di un fattore che può influenzare tanto la probabilità che un gioco venga notato e recensito su BGG, e di conseguenza la sua stessa presenza nei panieri del capitolo 4, quanto il numero di voti raccolti, che a sua volta influenza il *Geek Rating*. Le associazioni e le caratteristiche individuate in questo lavoro vanno quindi lette al netto di questa componente, che nessuna delle due analisi era in grado di isolare con i dati a disposizione.

6.4 – Sviluppi Futuri

Al di là dei limiti appena discussi, il lavoro suggerisce una possibile direzione di approfondimento futura: sottoporre a verifica sperimentale, per entrambi i capitoli, quanto osservato finora solo su base associativa.

Per il capitolo 4, la fase pilota già delineata al par. 4.9 – un banner o uno sconto limitato a poche coppie di titoli, una campagna e-mail rivolta a un segmento ridotto di clientela – costituirebbe un test controllato in grado di verificare se le associazioni individuate nelle recensioni si traducano effettivamente in un comportamento di acquisto congiunto, validando i risultati.

Per il capitolo 5, un test sperimentale come quello ipotizzato al par. 5.8.3 – sottoporre uno stesso titolo, con e senza una determinata meccanica o componente strutturale, a gruppi di giocatori – permetterebbe di verificare se le sette caratteristiche individuate abbiano effettivamente un ruolo causale nel determinare il gradimento di un gioco.

Bibliografia

- Agrawal, R., Imielinski, T., Swami, A. (1993). Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 207-216. <https://doi.org/10.1145/170035.170072>
- Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules in large databases. Proceeding of the 20th International Conference on Very Large Data Bases (VLDB '94), Santiago, Chile, 487-499.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32. <https://doi.org/10.1023/A:1010933404324>
- Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth International Group.
- Brin, S., Motwani, R., Ullman, J., & Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. *ACM SIGMOD Record*, 26(2), 255-264. <https://doi.org/10.1145/253262.253325>
- Gomez-Uribe, C. A., Hunt, N. (2015). The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information System*, 6 (4) 13, 1-19. <https://doi.org/10.1145/2843948>
- Hahsler, M., Grün, B., & Hornik, K. (2005). arules - A Computational Environment for Mining Association Rules and Frequent Item Sets. *Journal of Statistical Software*, 14(15), 1-25. <https://doi.org/10.18637/jss.v014.i15>
- Jaccard, P. (1912). The distribution of the flora in the alpine zone. *New Phytologist*, 11(2), 37-50. <https://doi.org/10.1111/j.1469-8137.1912.tb05611.x>
- Kamakura, W. A. (2007). Cross-Selling: Offering the Right Product to the Right Customer at the Right Time. *Journal of Relationship Marketing*, 6 (3-4), 41-58. https://doi.org/10.1300/J366v06n03_03

Naghikhani, S. (2024). Beyond The Box: A Comprehensive Market Research of The Board Game Industry.

Shmueli, G., Koppius, O. R. (2011). Predictive Analytics in Information Systems Research. *MIS Quarterly*, 35 (3), 553-572. <https://doi.org/10.2307/23042796>

Therneau, T. M., & Atkinson, E. J. (1997). An introduction to recursive partitioning using the RPART routines (Technical Report No. 61). Section of Biostatistics, Mayo Clinic, Rochester, Minnesota.

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6), 80-83. <https://doi.org/10.2307/3001968>

Wright, M. N., & Ziegler, A. (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, 77(1), 1-17. <https://doi.org/10.18637/jss.v077.i01>

Sitografia

Global Market Insights, 2025. Board Games Market Size & Share 2025 – 2034. Url: <https://www.gminsights.com/industry-analysis/board-games-market>.

Consultato ad Agosto 2026.

The NPD Group, 2020. The NPD Group Reports on First Half 2020 Global Toy Industry Sales. Url: <https://www.prweb.com/releases/the-npd-group-reports-on-first-half-2020-global-toy-industry-sales-818001466.html>. Consultato ad Agosto 2026.

DW.com, Matalucci Sergio, 2021. Board games sales growth faces pandemic challenges. DW.com, Url: <https://www.dw.com/en/coronavirus-rapid-growth-of-board-games-market-faces-pandemic-hurdles/a-56370700>. Consultato ad Agosto 2026.

Hasbro, 2021. Hasbro Reports Fourth Quarter and Full-Year 2020 Financial results. Url: <https://investor.hasbro.com/news-releases/news-release-details/hasbro-reports-fourth-quarter-and-full-year-2020-financial>. Consultato ad Agosto 2026.

RetailWire, Tom Ryan, 2021. Is the toy category's pandemic sales boon coming to an end? Url: <https://retailwire.com/discussion/is-the-toy-categorys-pandemic-sales-boon-coming-to-an-end/>. Consultato ad Agosto 2026.

Dianajonesaward.org, 2010. The 2010 Diana Jones Award for Excellence in Gaming has been given to BoardGameGeek.com. Url: <https://www.dianajonesaward.org/the-2010-award/>. Consultato ad Agosto 2026.

Tabletok.games, Stephen Lotters, 2025. BoardGameGeek Reaches 4,000,000 registered users. Url: <https://www.tabletok.games/blog/boardgamegeek-reaches-4000000-registered-users/>. Consultato ad Agosto 2026.

Appendice

Capitolo 2

A.2.1 - Tabella riassuntiva dei campi del dataset giochi

Campo	Descrizione
(senza nome)	Indice di riga residuo dall'esportazione (non è un campo informativo)
type	Tipo di elemento BGG (es. "boardgame")
id	Codice identificativo univoco del gioco su BGG
thumbnail	URL miniatura dell'immagine di copertina
image	URL immagine di copertina a risoluzione piena
alternate	Nomi alternativi/traduzioni del titolo
description	Descrizione testuale del gioco
yearpublished	Anno di pubblicazione
minplayers	Numero minimo di giocatori
maxplayers	Numero massimo di giocatori
suggested_num_players	Numero di giocatori consigliato secondo i voti della community
suggested_playerage	Età consigliata secondo i voti della community
suggested_language_dependence	Livello di dipendenza dalla lingua secondo i voti della community
playingtime	Durata di gioco dichiarata (minuti)
minplaytime	Durata minima dichiarata
maxplaytime	Durata massima dichiarata
minage	Età minima consigliata dall'editore
boardgamecategory	Lista delle categorie tematiche/di genere
boardgamemechanic	Lista delle meccaniche di gioco
boardgamefamily	Famiglia/serie editoriale di appartenenza
boardgameexpansion	Elenco delle espansioni collegate
boardgameaccessory	Elenco degli accessori collegati
boardgamecompilation	Eventuali compilation/raccolte a cui appartiene
boardgameimplementation	Collegamento ad altre edizioni/implementazioni dello stesso gioco
boardgamedesigner	Lista dei designer
boardgameartist	Lista degli illustratori
boardgamepublisher	Lista degli editori
usersrated	Numero di utenti che hanno votato il gioco

average	Voto medio grezzo
bayesaverage	Voto medio bayesiano (corretto per il numero di voti)
Board Game Rank	Posizione in classifica generale BGG
Strategy Game Rank	Posizione in classifica di categoria "Strategy"
Family Game Rank	Posizione in classifica di categoria "Family"
stddev	Deviazione standard dei voti
median	Voto mediano
owned	Numero di utenti che possiedono il gioco
trading	Numero di utenti disposti a scambiarlo
wanting	Numero di utenti che lo desiderano
wishing	Numero di utenti che lo hanno in wishlist
numcomments	Numero di commenti testuali
numweights	Numero di voti di complessità (peso) espressi
averageweight	Complessità media percepita (peso)
boardgameintegration	Collegamento ad altri giochi con cui è "integrabile"
Abstract Game Rank	Classifica di categoria "Abstract"
Party Game Rank	Classifica di categoria "Party"
Thematic Rank	Classifica di categoria "Thematic"
War Game Rank	Classifica di categoria "War"
Customizable Rank	Classifica di categoria "Customizable"
Children's Game Rank	Classifica di categoria "Children's"
RPG Item Rank	Classifica di categoria "RPG Item"
Accessory Rank	Classifica di categoria "Accessory"
name	Titolo del gioco

Capitolo 4

Cod. A.4.1 - Test di co-associazione di Jaccard tra le due edizioni (par. 4.2.4)

```
gruppi_confermati <- gruppi[riedizione_confermata == TRUE]
gruppi_da_unire <- copy(gruppi_confermati)
conteggio_reale <- recensioni[, .N, by = ID]
gruppi_da_unire[conteggio_reale, n_reale := i.N, on = "id==ID"]
gruppi_da_unire[is.na(n_reale), n_reale := 0]
gruppi_da_unire[, id_canonico := id[which.max(n_reale)], by = name_norm]
gruppi_da_controllare <- head(
  gruppi_da_unire[, .(n_reale_tot = sum(n_reale)), by = name_norm][order(-
    n_reale_tot)]$name_norm,
```

```

8)
recensioni_8 <- recensioni[rating >= 8]
test_jaccard <- function(pop_data, assoc_data, top_n = 10) {
  risultati <- data.table()
  for (g in gruppi_da_controllare) {
    ids_gruppo <- gruppi_da_unire[name_norm == g, id]
    id_vecchio_g <- gruppi_da_unire[name_norm == g & id != id_canonico, id]
    id_nuovo_g <- unique(gruppi_da_unire[name_norm == g, id_canonico])

    utenti_vecchio <- unique(pop_data[ID %in% id_vecchio_g, user])
    utenti_nuovo <- unique(pop_data[ID %in% id_nuovo_g, user])
    altri_vecchio <- assoc_data[user %in% utenti_vecchio & !ID %in% ids_gruppo,
      .N, by = ID][order(-N)][1:top_n, ID]
    altri_nuovo <- assoc_data[user %in% utenti_nuovo & !ID %in% ids_gruppo,
      .N, by = ID][order(-N)][1:top_n, ID]
    j <- length(intersect(altri_vecchio, altri_nuovo)) / length(union(altri_vecchio,
      altri_nuovo))
    risultati <- rbind(risultati, data.table(gruppo = g, jaccard = round(j, 2)))
  }
  risultati
}

risultati_totale <- test_jaccard(recensioni, recensioni)
risultati_rating8 <- test_jaccard(recensioni_8, recensioni_8)

```

Cod. A.4.2 - Verifica di assenza di informazione nei diversi campi del dataset Giochi (par. 4.2.5)

```

parse_lista <- function(x) {
  if (is.na(x) || x == "" || x == "[ ]") return(character(0))
  m <- gregexpr("'[^']*'|\"\"[^\"]*\"\"", x)
  matches <- regmatches(x, m)[[1]]
  if (length(matches) == 0) return(character(0))
  matches <- ifelse(substr(matches, 1, 2) == '\"\"',
    substr(matches, 3, nchar(matches)-2),
    substr(matches, 2, nchar(matches)-1))
}

```

```

matches[matches != ""]
}
parse_designer <- function(x) {
  v <- parse_lista(x)
  v[!v %in% c("(Uncredited)", "(Unknown)")]
}
sum(sapply(giochi$boardgamedesigner, function(x) length(parse_designer(x)) == 0))
sum(sapply(giochi$boardgamemechanic, function(x) length(parse_lista(x)) == 0))
sum(sapply(giochi$boardgamecategory, function(x) length(parse_lista(x)) == 0))
sum(sapply(giochi$boardgamefamily, function(x) length(parse_lista(x)) == 0))
sum(giochi$minage == 0); sum(giochi$playingtime == 0)
sum(giochi$minplayers == 0); sum(giochi$maxplayers == 0)
sum(giochi$maxplayers >= 99); sum(giochi$averageweight == 0)

```

Cod. A.4.3 - Confronto tra filtro rating $\geq$ 6 e rating $\geq$ 8 a parità di supporto e confidenza (par. 4.3)

```

trans_6 <- recensioni[rating >= 6, .(items = list(as.character(ID))), by = user]
regole_6 <- apriori(as(trans_6$items, "transactions"),
parameter = list(supp = 0.01, conf = 0.6, minlen = 2, maxlen = 3))
summary(quality(regole_6)$lift)

trans_8 <- recensioni[rating >= 8, .(items = list(as.character(ID))), by = user]
regole_8 <- apriori(as(trans_8$items, "transactions"),
parameter = list(supp = 0.01, conf = 0.6, minlen = 2, maxlen = 3))
summary(quality(regole_8)$lift)

```

Cod. A.4.4 - Esplorazione sistematica dei parametri di supporto e confidenza con l'applicazione del filtro rating $\geq$ 8 (par. 4.3)

```

trans_8_arules <- as(trans_8$items, "transactions")
supporti <- c(0.010, 0.011, 0.013)
confidenze <- c(0.5, 0.6, 0.7)
griglia <- data.table()
for (s in supporti) for (c in confidenze) {
  r <- apriori(trans_8_arules, parameter = list(supp = s, conf = c, minlen = 2, maxlen = 3))
  griglia <- rbind(griglia, data.table(supporto = s, confidenza = c, n_regole = length(r))) }

```

Cod. A.4.5 - Trasformazione delle regole in formato lungo e merge con le caratteristiche dei giochi (par. 4.4)

```
regole_tab <- as(regole, "data.frame")
setDT(regole_tab)
regole_tab[, rule_id := .I]

regole_tab[, lhs := gsub("[{}]", "", sub("=>.*", "", rules))]
regole_tab[, rhs := gsub("[{}]", "", sub(".*=> ", "", rules))]

lhs_long <- regole_tab[, .(id_char = trimws(unlist(strsplit(lhs, ",")))), by = rule_id]
lhs_long[, tipo := "antecedente"]
rhs_long <- regole_tab[, .(id_char = trimws(unlist(strsplit(rhs, ",")))), by = rule_id]
rhs_long[, tipo := "conseguente"]

regole_long <- rbind(lhs_long, rhs_long)
regole_long[, id := as.integer(id_char)]
regole_long[, id_char := NULL]

regole_long <- merge(regole_long, regole_tab[, .(rule_id, support, confidence, lift)],
by = "rule_id")
regole_long <- merge(
regole_long,
giochi[, .(id, name, boardgamecategory, boardgamemechanic,
boardgamedesigner, boardgamefamily, averageweight)],
by = "id", all.x = TRUE
)
```

Cod. A.4.6 – Costruzione dell'indicatore stesso_brand (par. 4.5)

```
parse_family_game <- function(x) { v <- parse_lista(x); v[startsWith(v, "Game:")] }
norm_nome <- function(x) trimws(tolower(x))

lhs_list <- LIST(lhs(regole)); rhs_list <- LIST(rhs(regole))
setkey(giochi, id)
n <- length(lhs_list)
stesso_brand <- logical(n)

for (i in seq_len(n)) {
ids_ant <- as.integer(lhs_list[[i]]); id_con <- as.integer(rhs_list[[i]])
righe_ant <- giochi[.(ids_ant)]; righe_con <- giochi[.(id_con)]
```

```

fam_con <- parse_family_game(riga_con$boardgamefamily)
fam_ant <- lapply(righe_ant$boardgamefamily, parse_family_game)
match_family <- any(sapply(fam_ant, function(f) length(intersect(f, fam_con)) > 0))

nome_con <- norm_nome(riga_con$name); nomi_ant <- norm_nome(righe_ant$name)
match_strutt <- FALSE
for (a in seq_along(ids_ant)) {
  for (campo in c("boardgameexpansion", "boardgamecompilation",
"boardgameimplementation")) {
    lista_ant <- norm_nome(parse_lista(righe_ant[[campo]][a]))
    lista_con <- norm_nome(parse_lista(riga_con[[campo]]))
    if (nome_con %in% lista_ant || nomi_ant[a] %in% lista_con) match_strutt <- TRUE
  }
}

stesso_brand[i] <- match_family || match_strutt
}

# controllo aggiuntivo: giochi coinvolti che appartengono a un gruppo di nomi duplicati
tutti_ids <- unique(as.integer(unlist(c(lhs_list, rhs_list))))
ids_in_dup_group <- tutti_ids[tutti_ids %in% gruppi_da_unire$id]
length(ids_in_dup_group)

for (id_target in ids_in_dup_group) {
  regole_coinvolte <- which(sapply(seq_len(n), function(i)
id_target %in% as.integer(lhs_list[[i]]) || id_target == as.integer(rhs_list[[i]])))
  print(regole_coinvolte)
}

```

Cod. A.4.7 – Verifica dei valori segnaposto tra i 68 giochi coinvolti nelle 324 regole (par. 4.5)

```

tutti_ids <- unique(as.integer(unlist(c(lhs_list, rhs_list))))
giochi_regole <- giochi[.(tutti_ids)]
sum(sapply(giochi_regole$boardgamemechanic, function(x) length(parse_lista(x)) == 0))
sum(sapply(giochi_regole$boardgamecategory, function(x) length(parse_lista(x)) == 0))
sum(sapply(giochi_regole$boardgamedesigner, function(x) length(parse_designer(x)) == 0))

```

Cod. A.4.8 – Calcolo degli indici di Jaccard e costruzione del dataset di feature (par. 4.5.2)

```
jaccard <- function(a, b) length(intersect(a, b)) / length(union(a, b))

regole_albero <- data.table(
  regola_id = 1:n, lift = quality(regole)$lift, support = quality(regole)$support,
  confidence = quality(regole)$confidence, stesso_brand = stesso_brand,
  mecc_max = numeric(n), mecc_min = numeric(n),
  cat_max = numeric(n), cat_min = numeric(n),
  designer_max = numeric(n), designer_min = numeric(n)
)

for (i in seq_len(n)) {
  ids_ant <- as.integer(lhs_list[[i]]); id_con <- as.integer(rhs_list[[i]])
  righe_ant <- giochi[.(ids_ant)]; riga_con <- giochi[.(id_con)]

  mecc_con <- parse_lista(riga_con$boardgamemechanic)
  cat_con <- parse_lista(riga_con$boardgamecategory)
  des_con <- parse_designer(riga_con$boardgamedesigner)

  j_mecc <- sapply(righe_ant$boardgamemechanic, function(x) jaccard(parse_lista(x),
    mecc_con))
  j_cat <- sapply(righe_ant$boardgamecategory, function(x) jaccard(parse_lista(x),
    cat_con))
  j_des <- sapply(righe_ant$boardgamedesigner, function(x) jaccard(parse_designer(x),
    des_con))

  regole_albero$mecc_max[i] <- max(j_mecc); regole_albero$mecc_min[i] <- min(j_mecc)
  regole_albero$cat_max[i] <- max(j_cat); regole_albero$cat_min[i] <- min(j_cat)
  regole_albero$designer_max[i] <- max(j_des); regole_albero$designer_min[i] <- min(j_des)
}

jaccard_cols <- c("mecc_max", "mecc_min", "cat_max", "cat_min", "designer_max",
  "designer_min")
all(sapply(regole_albero[, ..jaccard_cols], function(x) all(x >= 0 & x <= 1)))
sum(is.na(regole_albero))
```

Cod. A.4.9 – Verifica di stabilità dell'albero al variare di minbucket (par. 4.7)

```
for (mb in 10:35) {
  set.seed(123)
  a <- rpart(lift ~ mecc_max + mecc_min + cat_max + cat_min + designer_max + designer_min
```

```

+ stesso_brand,
data = regole_albero, method = "anova",
control = rpart.control(minbucket = mb, cp = 0.001))

tabella_cp <- a$cptable
xerror_min <- min(tabella_cp[, "xerror"])
xstd_min <- tabella_cp[which.min(tabella_cp[, "xerror"]), "xstd"]
riga_1se <- min(which(tabella_cp[, "xerror"] <= xerror_min + xstd_min))
a_pot <- prune(a, cp = tabella_cp[riga_1se, "CP"])

radice <- as.character(a_pot$frame$var[1])
n_split <- sum(a_pot$frame$var != "<leaf>")
imp <- a_pot$variable.importance
imp_pct <- round(100*imp/sum(imp), 1)

cat(mb, radice, n_split, imp_pct[["mecc_max"]], imp_pct[["designer_max"]],
tabella_cp[riga_1se, "xerror"], tabella_cp[riga_1se, "xstd"], "\n")
}

```

Codice A.4.10 – Rappresentazione grafica dell'albero di regressione (Fig.4.1, par. 4.7)

```

library(rpart.plot)
# albero_potato costruito come al par. 4.7 (set.seed(123), minbucket = 15)
rpart.plot(albero_potato,
type = 2, extra = 101, under = TRUE,
fallen.leaves = TRUE, faclen = 0,
box.palette = "Blues", shadow.col = "gray85",
branch = 0.4, tweak = 1.3, digits = 3,
split.font = 2, family = "sans")

```

Codice A.4.11 – Calcolo del lift standardizzato e media per foglia (par. 4.7)

```

sigma <- 0.010; kappa <- 0.6
info(regole)$support <- sigma
info(regole)$confidence <- kappa
regole_albero[, lift_std := interestMeasure(regole, "stdLift", transactions =
trans_8_arules, correct = TRUE)]

regole_albero[, foglia := albero_potato$where]
regole_albero[, .(n = .N, lift_medio = mean(lift), lift_std_medio = mean(lift_std)), by
= foglia]

```

Codice A.4.12 - Verifica del valore di designer_max e calcolo dell'importanza dell'indicatore stesso_brand (par. 4.8)

```
foglia_strutturale <- regole_albero[mecc_max >= 0.7667]
summary(foglia_strutturale$designer_max)

importanza <- albero_potato$variable.importance
round(100 * importanza["stesso_brand"] / sum(importanza), 1)
```

Codice A.4.13 – Meccaniche e categorie più ricorrenti nella foglia 1 (par. 4.8)

```
idx_foglia <- which(regole_albero$mecc_max >= 0.169 & regole_albero$mecc_max < 0.7667)
ids_foglia <- unique(as.integer(unlist(c(lhs_list[idx_foglia], rhs_list[idx_foglia]))))
sub_g <- giochi[id %in% ids_foglia]
mecc_all <- unlist(lapply(sub_g$boardgamemechanic, parse_lista))
cat_all <- unlist(lapply(sub_g$boardgamecategory, parse_lista))
sort(table(mecc_all), decreasing = TRUE)
sort(table(cat_all), decreasing = TRUE)
```

Codice A.4.14 – Confronto di profilo tramite test di Wilcoxon Foglia 1 (par. 4.8)

```
sub_g <- giochi[id %in% ids_foglia]
ids_68 <- unique(as.integer(unlist(c(lhs_list, rhs_list))))
riferimento <- giochi[!id %in% ids_68]
wilcox.test(sub_g$averageweight[sub_g$averageweight > 0],
riferimento$averageweight[riferimento$averageweight > 0])
wilcox.test(sub_g$playingtime[sub_g$playingtime > 0],
riferimento$playingtime[riferimento$playingtime > 0])
wilcox.test(sub_g$minage[sub_g$minage > 0], riferimento$minage[riferimento$minage > 0])
```

Codice A.4.15 – Meccaniche e categorie più ricorrenti nella foglia 2 (par. 4.8)

```
idx_foglia <- which(regole_albero$mecc_max < 0.169)
ids_foglia <- unique(as.integer(unlist(c(lhs_list[idx_foglia], rhs_list[idx_foglia]))))
sub_g <- giochi[id %in% ids_foglia]
mecc_all <- unlist(lapply(sub_g$boardgamemechanic, parse_lista))
cat_all <- unlist(lapply(sub_g$boardgamecategory, parse_lista))
sort(table(mecc_all), decreasing = TRUE)
sort(table(cat_all), decreasing = TRUE)
```

Codice A.4.16 – Confronto di profilo tramite test di Wilcoxon Foglia 2 (par. 4.8)

```
sub_g <- giochi[id %in% ids_foglia]
```

```
wilcox.test(sub_g$averageweight[sub_g$averageweight > 0],
riferimento$averageweight[riferimento$averageweight > 0])
wilcox.test(sub_g$playingtime[sub_g$playingtime > 0],
riferimento$playingtime[riferimento$playingtime > 0])
wilcox.test(sub_g$minage[sub_g$minage > 0], riferimento$minage[riferimento$minage > 0])
```

Capitolo 5

Codice A.5.1 – Verifica di valori mancanti e segnaposto nel dataset giochi (par. 5.1)

```
# controllo di celle realmente vuote (NA) nei campi coinvolti
colSums(is.na(giochi[, .(minage, playingtime, minplayers, maxplayers,
                        boardgamecategory, boardgamemechanic, boardgamedesigner)]))

# conteggio delle condizioni segnaposto nei campi numerici (Tab. 5.1)
sum(giochi$minage == 0)
sum(giochi$playingtime == 0)
sum(giochi$minplayers == 0)
sum(giochi$maxplayers == 0)
sum(giochi$maxplayers >= 99)

# conteggio delle condizioni segnaposto nei campi testuali (Tab. 5.1)
# funzioni parse_lista() e parse_designer() già definite in Appendice A.4.2
sum(sapply(giochi$boardgamemechanic, function(x) length(parse_lista(x)) == 0))
sum(sapply(giochi$boardgamecategory, function(x) length(parse_lista(x)) == 0))
sum(sapply(giochi$boardgamedesigner, function(x) length(parse_designer(x)) == 0))
```

Codice A.5.2 – Trattamento valori segnaposto e costruzione variabili dummy (par. 5.2)

```
nome_pulito <- function(x) {
  x <- gsub("[^A-Za-z0-9]+", "_", x)
  gsub("^_|_+$", "", x)
}

dummy_val <- function(l, val) {
  if (length(l) == 0) return(NA_integer_)
  as.integer(val %in% l)
}
```

```

}
mecc_liste <- lapply(giochi$boardgamemechanic, parse_lista)
tutte_mecc <- unique(unlist(mecc_liste))
for (m in tutte_mecc) {
  col <- paste0("mecc_", nome_pulito(m))
  giochi[[col]] <- sapply(mecc_liste, dummy_val, val = m)
}
cat_liste <- lapply(giochi$boardgamecategory, parse_lista)
tutte_cat <- unique(unlist(cat_liste))
for (c in tutte_cat) {
  col <- paste0("cat_", nome_pulito(c))
  giochi[[col]] <- sapply(cat_liste, dummy_val, val = c)
}

```

Codice A.5.3 – Cardinalità del campo designer, costruzione di designer_exp e controllo di sanità (par. 5.2.2)

```

# cardinalità del campo designer
des_all <- unlist(lapply(giochi$boardgamedesigner, parse_designer))
length(unique(des_all))
freq_des <- table(des_all)
mean(freq_des == 1) * 100

# costruzione di designer_exp
des_liste <- lapply(giochi$boardgamedesigner, parse_designer)
n_per_game <- sapply(des_liste, length)
dt_long <- data.table(
  id = rep(giochi$id, n_per_game),
  anno = rep(giochi$yearpublished, n_per_game),
  designer = unlist(des_liste)
)
dt_long[, esperienza := sapply(anno, function(a) sum(anno < a)), by = designer]
exp_max <- dt_long[, .(designer_exp = max(esperienza)), by = id]
giochi <- merge(giochi, exp_max, by = "id", all.x = TRUE)

```

```
# i giochi senza designer valido restano NA: nessuna riga di fill a 0

# controllo di sanità: Reiner Knizia
kn <- giochi[grepl("Reiner Knizia", boardgamedesigner, fixed =
TRUE)][order(yearpublished)]
nrow(kn); kn$name[1]; kn$yearpublished[1]; kn$designer_exp[1]
!is.unsorted(kn$designer_exp, na.rm = TRUE)
```

Codice A.5.4 – Scansione dei valori di minbucket (par. 5.5.2)

```
# --- Tab. 5.2: dimensione dell'albero potato ed errore CV al variare di minbucket
(seed=123) ---

valori_mb <- c(3, 5, 10, 20, 30, 40, 50, 75, 100)
tabella_minbucket <- data.table()
for (mb in valori_mb) {
  set.seed(123)
  a <- rpart(bayesaverage ~ ., data = train_model, method = "anova", minbucket = mb, cp
= 0.001)
  cpt <- a$cptable
  min_xerror <- min(cpt[, "xerror"])
  se_min <- cpt[which.min(cpt[, "xerror"]), "xstd"]
  soglia <- min_xerror + se_min
  riga <- which(cpt[, "xerror"] <= soglia)[1]
  albero <- prune(a, cp = cpt[riga, "CP"])
  foglie <- albero$frame[albero$frame$var == "<leaf>", ]
  tabella_minbucket <- rbind(tabella_minbucket, data.table(
    minbucket = mb, n_split = cpt[riga, "nsplit"], n_foglie = nrow(foglie),
    oss_minime_foglia = min(foglie$n), xerror = round(min_xerror, 4)
  ))
}
tabella_minbucket
```

Codice A.5.5 – Costruzione dell'albero (par. 5.5.3)

```
set.seed(123)
```

```

albero_full <- rpart(bayesaverage ~ ., data = train_model, method = "anova", minbucket =
20, cp = 0.001)
cpt <- albero_full$cptable
min_xerror <- min(cpt[, "xerror"])
se_min <- cpt[which.min(cpt[, "xerror"]), "xstd"]
soglia <- min_xerror + se_min
riga <- which(cpt[, "xerror"] <= soglia)[1]
albero <- prune(albero_full, cp = cpt[riga, "CP"])

```

Codice A.5.6 – Rappresentazione grafica dei primi tre livelli dell'albero Fig. 5.1 (par. 5.5.3)

```

set.seed(123)
alb_illustrativo <- rpart(bayesaverage ~ ., data = train_model, method = "anova",
                          minbucket = 20, cp = 0.001, maxdepth = 3)
rpart.plot(alb_illustrativo, type = 2, extra = 101, box.palette = "Blues",
           shadow.col = "gray", nn = FALSE, digits = 3)

```

Codice A.5.7 – Confronto tra chi ha la meccanica radice e chi non ce l'ha (par. 5.5.3)

```

# confronto generale sul training
media_con <- mean(train$bayesaverage[train$mecc_End_Game_Bonuses == 1])
media_senza <- mean(train$bayesaverage[train$mecc_End_Game_Bonuses == 0])
n_con <- sum(train$mecc_End_Game_Bonuses == 1)
n_senza <- sum(train$mecc_End_Game_Bonuses == 0)
# controllo di stabilita' temporale, interamente entro il training
train[, periodo := ifelse(yearpublished <= 2017, "fino al 2017", "2018-2020")]
train[, .(
  media_con = mean(bayesaverage[mecc_End_Game_Bonuses == 1]),
  media_senza = mean(bayesaverage[mecc_End_Game_Bonuses == 0])
), by = periodo]

```

Codice A.5.8 - Calcolo dei valori dell'RMSE ed esempio: Noli (par. 5.5.4)

```

# previsioni e RMSE
pred_train <- predict(albero, train_model)

```

```

pred_test <- predict(albero, test_model)
rmse_train <- sqrt(mean((train_model$bayesaverage - pred_train)^2))
rmse_test <- sqrt(mean((test_model$bayesaverage - pred_test)^2))

media_train <- mean(train_model$bayesaverage)
rmse_baseline <- sqrt(mean((test_model$bayesaverage - media_train)^2))

# esempio: Noli
idx_noli <- which(test$name == "Noli")
test$name[idx_noli]
test$bayesaverage[idx_noli]
media_train
pred_test[idx_noli]

```

Dichiarazione del Ricorso a Strumenti di Intelligenza Artificiale (IA)

Il sottoscritto Mattia Vignali, matricola numero 908698, iscritto presso Cà Foscari al corso di Laurea magistrale in Marketing e Comunicazione, autore della tesi di laurea Data Mining e Analisi Predittiva nel Mercato dei Giochi da Tavolo, relatrice Daniela Favaretto, consapevole delle sanzioni previste dall'art. 76 del Testo unico, D.P.R. 28/12/2000 n.445, e della decadenza dei benefici prevista dall'art. 75 del medesimo Testo unico in caso di dichiarazioni false o mendaci, sotto la propria personale responsabilità ai sensi degli artt. 46 e 47 del D.P.R. 28/12/2000 n. 445,

DICHIARA

- ☐ di NON aver fatto ricorso a tecnologie di intelligenza artificiale generativa per supportare vari aspetti della ricerca e della scrittura della sua tesi di laurea;
- ☒ di aver fatto ricorso a tecnologie di intelligenza artificiale generativa per supportare vari aspetti della ricerca e della scrittura della sua tesi di laurea, in particolare:
 - Claude Sonnet 5, sviluppato da Anthropic – rilasciato il 30 Giugno 2026
 - Utilizzato per:
 - Supporto allo svolgimento dell'analisi sui dati, in particolare per la scrittura dei codici R.
 - Revisione dei testi per migliorare chiarezza e coerenza
 - Individuazione delle fonti
 - Per quanto riguarda la scrittura dei codici e il supporto all'analisi questi sono circoscritti ai capitoli 4 e 5. Il supporto nell'individuazione delle fonti è invece riferito ai capitoli 1-3. La revisione dei testi è relativa a diversi stadi del presente lavoro a seconda della necessità.
 - I codici R forniti e tutti i risultati ottenuti sono stati verificati nell'ambiente R, le fonti individuate sono state analizzate dall'autore

prima di essere prese in considerazione e inserite nella tesi. Nessuna frase revisionata dallo strumento di Intelligenza Artificiale è stata considerata valida senza prima essere controllata dall'autore.

DICHIARA INOLTRE

che tutto il materiale generato dall'IA è stato attentamente verificato, rivisto e rielaborato dall'autore, che si assume la piena responsabilità per la qualità, l'accuratezza e l'integrità scientifica del lavoro.