ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

ANALISI E PROTOTIPAZIONE DI UN SISTEMA PER LA PREVISIONE DELLE VENDITE DI UN BRAND DI ABBIGLIAMENTO

Elaborato in
Data Mining

Relatore
Prof. Matteo Golfarelli

Presentata da
Giacomo Montalti

Quarta Sessione di Laurea
Anno Accademico 2019 – 2020

PAROLE CHIAVE

Clustering

Forecasting

Planning

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Negli ultimi anni le principali innovazioni tecnologiche a livello aziendale hanno interessato la macro-disciplina del data mining, ovvero l'elaborazione dei dati generati a qualunque livello dai diversi processi aziendali. Sono aumentate le disponibilità in termini di capacità computazionali e di memorizzazione, qualunque tipo di attività crea costantemente nuovi dati, in ogni momento della giornata: registratori di cassa, videocamere di videosorveglianza e sensori utilizzati nei più disparati ambiti. Tutto questo ha portato a un trend tecnologico fortemente incentrato sull'*informazione*, intesa come prezioso asset con cui poter ottenere un vantaggio determinante sui concorrenti. Per poter estrarre informazioni utili dai dati grezzi si utilizzano diversi approcci, appartenenti a categorie di tecniche automatiche o semi-automatiche, con lo scopo di analizzare i dati e ottenere il miglior risultato possibile da essi. Per poter conseguire dei risultati soddisfacenti si utilizzano, generalmente, approcci composti da diverse fasi: analisi del dato, preprocessing ed elaborazione, modellazione e sviluppo di statistiche di interesse.

All'interno di questa tesi verranno applicate le principali tecniche di data mining con lo scopo di ricavare informazioni utili dai dati riguardanti un brand di abbigliamento fast fashion: il marchio Terranova del Gruppo Teddy. Anche all'interno di questa realtà aziendale i dati hanno un ruolo fondamentale: diverse fasi (dalla logistica all'organizzazione del personale) sono guidate interamente da ciò che i diversi sistemi informativi del brand registrano, elaborano e inoltrano ai database aziendali. Poiché uno dei principali obiettivi del gruppo è sempre stato quello di innovare i propri processi produttivi, il reparto *Information Technology* ha richiesto lo studio di nuovi approcci per l'ottimizzazione di alcune fasi aziendali. Per questo motivo, all'interno della tesi verranno presi in esame alcuni dei dati del gruppo Teddy relativi ai negozi e all'andamento delle vendite passate, con lo scopo di esplorare e applicare alcune delle tecniche di data mining esistenti. Verranno esplorati i diversi algoritmi di clustering, con lo scopo di verificare l'effettiva esistenza di gruppi coesi di negozi ben separati gli uni dagli altri. Si procederà con il forecasting delle vendite tramite modelli previsionali, descrivendo i vari approcci esistenti e le differenze che intercorrono tra gli algoritmi impiegati. Verranno infine offerti diversi strumenti per poter

analizzare la bontà dei risultati ottenuti.

La tesi è composta da quattro parti: prima di tutto verrà descritta all'interno del capitolo 1 la realtà aziendale, i motivi che hanno portato a questa tesi e lo stato attuale dei processi informativi che verranno successivamente analizzati. Si descriveranno inoltre le dinamiche aziendali più importanti e gli obiettivi fissati. Successivamente, all'interno del capitolo 2 verranno presentate le varie tecniche di data mining impiegate. Nella sezione 2.1 si approfondiranno nel dettaglio diversi approcci per il clustering, una famiglia di metodi utili a raggruppare elementi omogenei presenti all'interno di un insieme di dati, analizzando nello specifico pregi e difetti di ogni tecnica presa in esame. In seconda battuta, all'interno della sezione 2.2 verranno analizzati i più recenti metodi di previsione per serie storiche, confrontandone tra loro le caratteristiche e definendone casi d'uso ideali e consigliati. Successivamente, all'interno del capitolo 3, verranno analizzati nel dettaglio i dati forniti dal Gruppo Teddy, si descriveranno le strutture dei dataset e le varie fasi di preprocessing dei singoli attributi. Dopo aver definito le forme finali dei dati, si procederà all'interno del capitolo 4 a descrivere i risultati conseguiti, definendo quali modelli abbiano raggiunto le migliori prestazioni. Verranno inoltre confrontati le differenze tra i sistemi analizzati all'interno della tesi e quelli attualmente utilizzati dal Gruppo Teddy per sopperire alle richieste aziendali, così da avere un metro di confronto su ogni aspetto preso in esame.

Capitolo 1

Descrizione del background aziendale

All'interno di questo capitolo si descrivono le motivazioni che hanno portato alla proposta di tesi da parte del gruppo Teddy S.p.A., azienda italiana a capo di diversi marchi di moda, operante a livello globale nel settore del fast fashion.

Verrà dapprima fornita una descrizione della struttura organizzativa aziendale e dei punti vendita, per poi passare alle dinamiche dei sistemi informativi che hanno richiesto un'analisi approfondita. Si elencheranno, inoltre, gli obiettivi fissati del progetto svolto.

1.1 Il gruppo Teddy S.p.A

Il gruppo Teddy nasce a Riccione nel 1961, inizialmente come semplice negozio di abbigliamento e relativo laboratorio tessile artigianale a Rimini. In pochi anni subisce una forte crescita economica e un incremento delle vendite importante, che portano alle prime aperture di nuovi negozi. Nel giro di dieci anni l'espansione commerciale al di fuori dei confini provinciali e regionali comporta l'apertura di ulteriori negozi e l'inizio della vendita all'ingrosso in tutta Italia. Nel 1988 nasce il primo marchio di Teddy, ovvero *Terranova*, che in soli dieci anni sviluppa la sua presenza in ben venti stati diversi nel mondo, e in poco più di quindici anni porta il numero di negozi disponibili sul territorio a più di quattrocento. Nei primi anni 2000 vengono fondati ulteriori nuovi marchi di abbigliamento, tra cui *Rinascimento* e *Calliope*: i nuovi punti vendita aperti in giro per l'Italia favoriscono un periodo di crescita economica che aumenta anno dopo anno il fatturato del gruppo, il cui andamento è rappresentato in figura 1.1.

Ai tre brand già citati se ne aggiungono altri due: *Katana* e *QB24*. Al contrario dei precedenti, entrambi non dispongono di punti vendita dedicati,

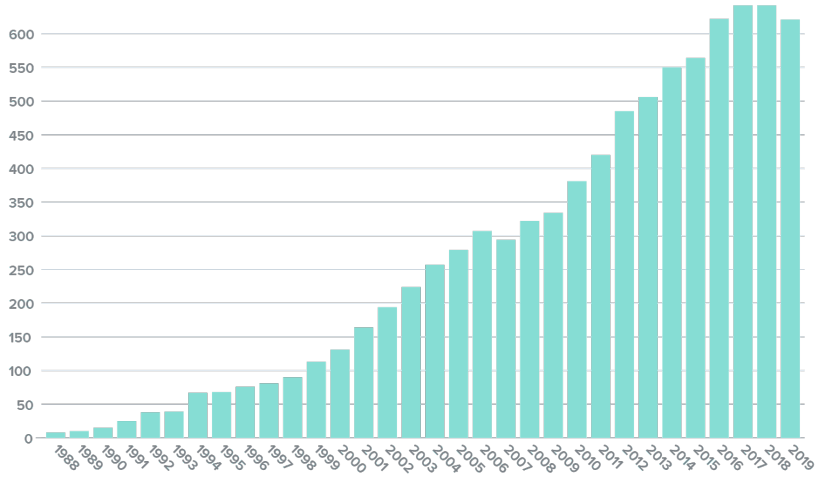


Figura 1.1: Fatturato del gruppo Teddy dal 1988 al 2019 (in milioni di euro).

ma la loro distribuzione è affidata a grossisti terzi o a spazi dedicati all'interno dei negozi del gruppo.

Ad oggi, Teddy può contare quasi **700 punti vendita**, dislocati in venti paesi differenti: 504 negozi *Terranova*, 123 shop *Calliope* e 61 del brand *Rinascimento*.

1.2 I sistemi informativi utilizzati

Un'azienda di dimensioni importanti come quelle del gruppo Teddy deve necessariamente appoggiarsi a sofisticati sistemi decisionali in grado di aiutare il management a prendere le decisioni giuste nei vari processi produttivi. Sfortunatamente, l'intera topologia dei sistemi impiegati dal gruppo per pianificare il lavoro è troppo complessa e articolata per essere descritta all'interno di questa tesi.

Per questo motivo, ci si focalizzerà principalmente su una particolare infrastruttura cruciale per l'organizzazione dei punti vendita: un sistema Microsoft SQL Server Analysis Services. Questo modulo software viene utilizzato dal 2013 per la previsione delle vendite all'interno dei singoli punti vendita. L'algoritmo utilizzato è chiamato **Microsoft Neural Network**. Può tranquillamente essere considerato il fulcro della fase organizzativa dei punti vendita, in quanto le

previsioni sviluppate da questo sistema vengono utilizzate principalmente per due attività aziendali essenziali:

- Il trasporto dei capi d'abbigliamento dal magazzino principale ai punti vendita;
- L'organizzazione dei personale nei singoli negozi.

Si procede descrivendo il flusso informativo che genera un punto vendita, in che modo questo si interfacci con il sistema Microsoft appena descritto e cosa venga generato da queste interazioni. Successivamente, si descriverà il funzionamento **pratico** del sistema e di come questo venga utilizzato dallo store manager (ovvero il responsabile del punto vendita) per definire i turni di lavoro ogni settimana.

1.2.1 Il flusso informativo generato dalle vendite

Ogni negozio è fornito di un gestionale che tiene conto delle vendite effettuate dai registri di cassa, e le mantiene in memoria. Nel corso della giornata inoltra i dati delle vendite effettuate (codice a barre dell'articolo e taglia) al sistema SQL Server Analysis Services[1] che le memorizza. La sincronizzazione tra dati del negozio e server remoto avviene più volte nell'arco della giornata lavorativa. In fase di chiusura casse le vendite vengono definitivamente sincronizzate con il server remoto. L'intera sequenza di azioni è rappresentata all'interno della figura 1.2.

1.2.2 Il forecast delle vendite tramite Microsoft Neural Network

Dopo aver discusso di come vengono generati e salvati i dati relativi alle vendite, all'interno di questa sottosezione si descrive in che modo vengano utilizzati per generare previsioni di vendita utili ai fini organizzativi.

Lo store manager necessita in maniera ciclica di conoscere le stime di vendita relative ai giorni successivi per organizzare i turni dei dipendenti. In linea di massima, generalmente per la settimana successiva a quella in corso. Esistono diverse mansioni all'interno di ogni punto vendita: stoccaggio, riordino, riassortimento, attività di cassa, presidio visivo e, per il momento, assistenza alla vendita. È quindi importante che le stime siano precise in modo da

[1] Tecnicamente il flusso informativo potrebbe essere più complesso e coinvolgere ulteriori sistemi informatici incaricati di svolgere differenti elaborazioni, ma per i fini della tesi è sufficiente sapere che SQL Server riceve i dati in questo modo.

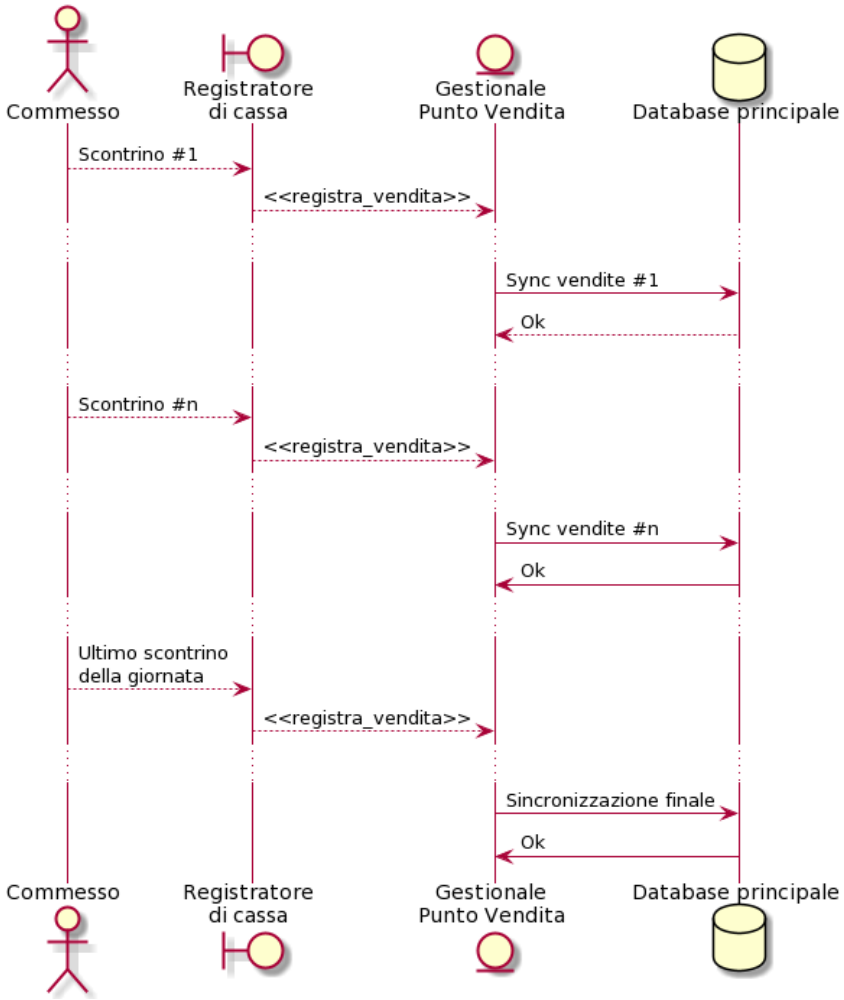


Figura 1.2: Diagramma di sequenza delle vendite nei negozi.

garantire un'organizzazione ottimale del punto vendita. Dopo aver impostato una previsione di forecast per il proprio negozio, l'interfaccia grafica visualizzata dallo store manager rappresentata in figura 1.3 include le seguenti informazioni:

- Nella **prima** riga le informazioni relative alla stessa settimana in corso, ma dell'anno precedente, sulla **seconda** riga le stesse informazioni ma relative alla settimana precedente a quella in corso, mentre nella terza riga le informazioni della settimana attuale;
- Per ogni giornata, la quantità di **vendite effettuate** (con il nome di *Consultivo*);
- Accanto allo storico di vendita, è disponibile una grafica con la relativa previsione meteorologica e la temperatura massima e minima;

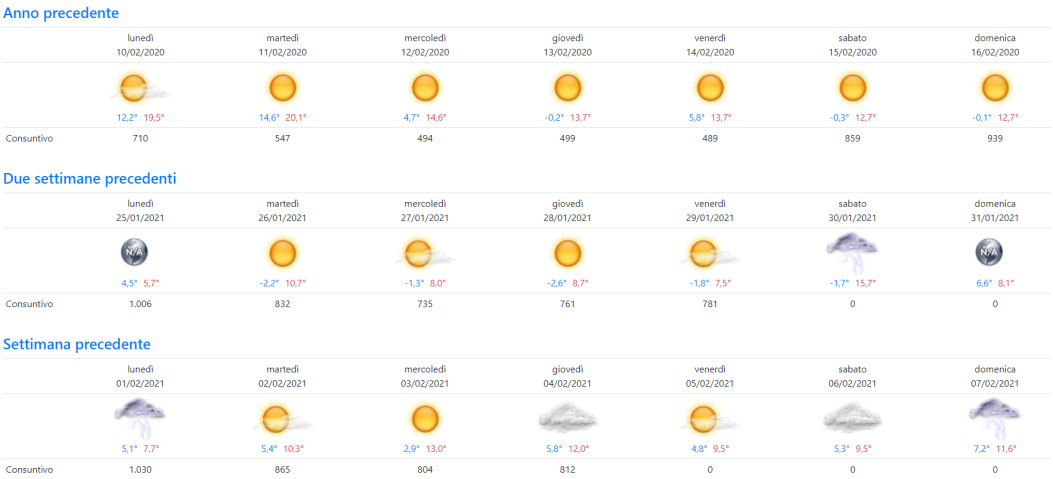


Figura 1.3: Storico delle vendite tipico tramite Microsoft Neural Network.

Se la richiesta di forecast viene eseguita a metà settimana, nella sezione *Settimana precedente* tutte le caselle relative a giorni futuri contengono un valore "0".

Subito sotto la sezione informativa, sono presenti le previsioni di vendita per la settimana successiva a quella in corso (rappresentate in figura 1.4), con un arrangiamento grafico molto simile a quello appena descritto.

L'unica differenza con la sezione storica, è la casella di input prevista sotto a ogni previsione, chiamata **rettifica**: lo store manager può inserire, giorno per giorno, quanti pezzi sono stati effettivamente venduti.

Il diagramma di sequenza completo delle interazioni tra responsabile punto vendita e algoritmo di previsione è rappresentato in figura 1.5. Nella fase di

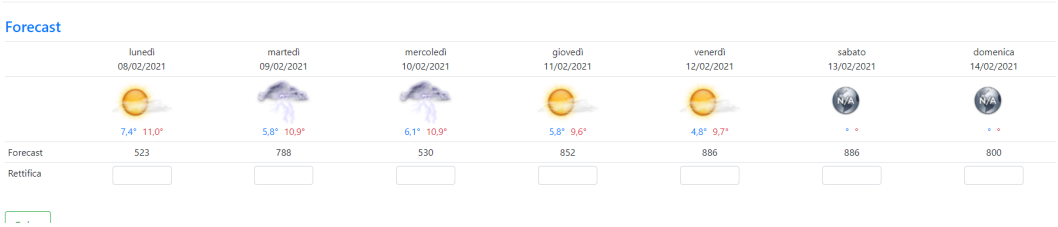


Figura 1.4: Forecast delle vendite tramite Microsoft Neural Network.

organizzazione del punto vendita lo store manager deve organizzare i turni dei dipendenti adeguati al carico di lavoro previsto, ma questo specifico aspetto verrà approfondito meglio nella sezione 1.2.4.

1.2.3 Fornitura della merce ai punti vendita

Le previsioni che vengono fornite dall'algoritmo sono utilizzate anche dal sistema di logistica aziendale per **l'organizzazione dei rifornimenti e delle spedizioni**. La gestione delle scorte dei punti vendita è particolarmente critica per un brand fast fashion, che deve cercare di aumentare i profitti puntando principalmente sulla quantità dei capi venduti. È necessario riassortire i prodotti prima che vengano esaurite le scorte ma al contempo senza saturare la capienza ridotta dei magazzini presenti nei punti vendita. Inoltre, per rimanere al passo con le esigenze dei clienti negli anni sono stati sviluppati nuovi canali di vendita:

- E-commerce web ed e-shop su applicazioni mobile;
- Click & Collect;
- Reserve & Collect.

E anche gli ordini eseguiti tramite questi canali devono essere correttamente gestiti ed evasi in tempi relativamente brevi. Per questi motivi, il brand Terranova **utilizza un unico grande centro di distribuzione** in grado di svolgere la funzione di magazzino per i rifornimenti dei punti vendita in Italia e sopperire ai diversi tipi di ordine. Inoltre, il **riordino è stato completamente automatizzato**: i responsabili di ogni negozio non devono ciclicamente occuparsi della merce mancante, poiché il sistema informativo su cui l'organizzazione dei negozi poggia conosce esattamente quali sono i capi e le taglie vendute. In questo modo, il punto vendita non si deve preoccupare di doversi procurare la merce da vendere, così come avviene in tante altre realtà aziendali, ma sarà il sistema a svolgere questa complessa funzione.

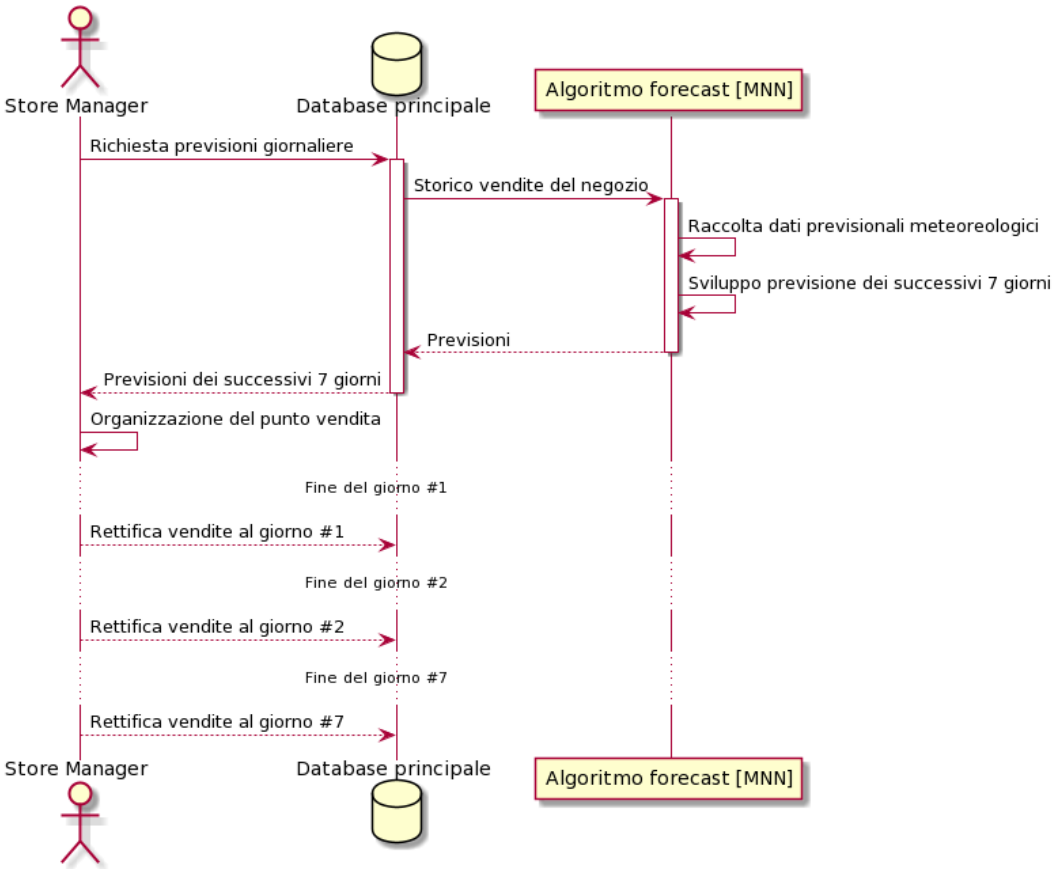


Figura 1.5: Diagramma di sequenza per il forecast con Microsoft Neural Network.

1.2.4 Turnazione dei dipendenti in negozio

Quello di cui deve effettivamente occuparsi lo store planner è la gestione dei turni lavorativi dei dipendenti. Come già descritto in precedenza, Microsoft Neural Network fornisce solo una stima di quanti pezzi verranno venduti, ma l'organizzazione dei turni di lavoro ricade completamente sul responsabile che con congruo anticipo deve definire gli orari di lavoro di ciascun dipendente. Per questo motivo, la stima che l'algoritmo genera deve cercare di essere il più possibile precisa, e al responsabile del negozio è richiesto di utilizzare l'esperienza acquisita negli anni per rimediare all'errore generato dal modello nelle previsioni di vendita. Nei momenti di maggiore richiesta, il rischio per Teddy è quello di avere situazioni con personale non sufficiente a rispondere alle richieste del punto vendita, limitando le vendite a quanti capi riescono ad essere riassortiti. Nel caso contrario, la presenza di troppi addetti presenti nel punto vendita farebbe lievitare i costi per il personale, riducendo i guadagni derivanti da sconti e promozioni disponibili e vanificando gli sforzi fatti per la fornitura della merce a livello logistico e organizzativo.

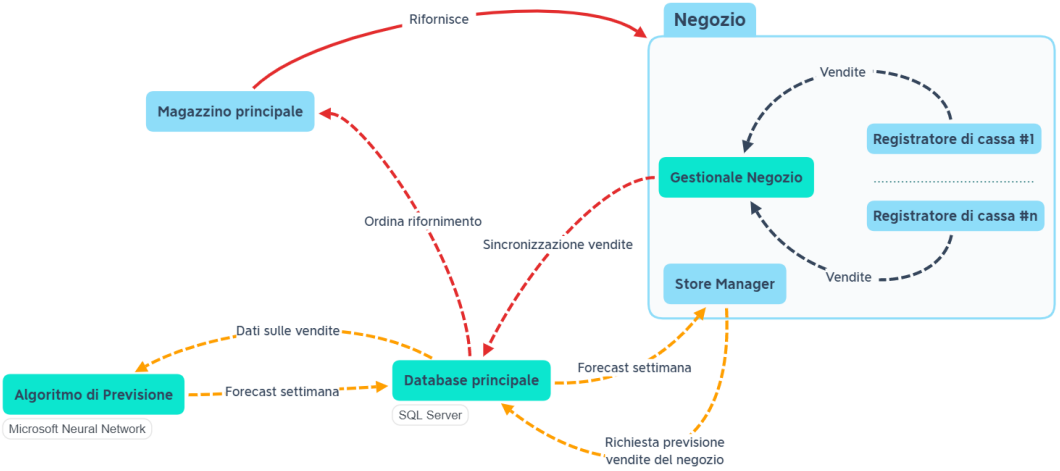


Figura 1.6: Rappresentazione riassuntiva dei flussi informativi tra le varie entità.

1.3 Gli obiettivi del progetto svolto

Dopo aver descritto lo stato attuale dei sistemi informativi aziendali del gruppo Teddy vengono definiti gli obiettivi del progetto di tesi. Il focus principale è stato incentrato sulla fase di previsione delle vendite, in particolare su due macro compiti:

- **La clusterizzazione** dei punti vendita più simili tra loro eseguendo diverse prove con i dati a disposizione di Teddy;
- L'identificazione di un metodo alternativo e parallelo per la **La previsione delle vendite** settimanali del numero di capi per ogni negozio.

1.3.1 Clustering dei negozi

Il raggruppamento dei negozi è stato richiesto dal gruppo Teddy principalmente per un motivo: i nuovi punti vendita appena aperti o future nuove aperture non hanno, ovviamente, alcuno storico del venduto. Di conseguenza, non è possibile sapere come organizzare i turni di lavoro. Inoltre, Microsoft Neural Network per poter sviluppare previsioni precise ha bisogno di una quantità di dati particolarmente consistente: **nei primi due anni il forecast non è particolarmente preciso**. Elaborare insiemi di punti vendita somiglianti a uno specifico negozio appena aperto o con poche settimane di storico può aiutare a sviluppare le previsioni di vendita, associandogli un andamento "simile" a quello dei negozi disponibili. In questo modo, si potrebbe soprassedere a quello che è il più grande difetto di un algoritmo di tipo neurale.

1.3.2 Metodo alternativo di previsione delle vendite

Microsoft Neural Network è un algoritmo estremamente preciso, richiede molti dati storici per poter svolgere le proprie previsioni, ma in linea di massima è in funzione sui sistemi del gruppo Teddy dal 2013, e ha svolto correttamente le proprie mansioni fino ad oggi. Sono stati, però, individuati alcuni difetti:

- È estremamente sensibile alle variabili ambientali come meteo e temperatura esterna (è per questo motivo che sono presenti in fase di previsione, come descritto all'interno della sezione 1.2.2);
- Non permette di considerare ulteriori variabili esterne ritenute importanti, come le varie situazioni che portano ad un aumento dei pezzi venduti (saldi, vendita promozionale, promozioni target tramite fidelity card);
- Dall'esperienza acquisita durante gli anni ci si è accorti che, per motivi puramente organizzativi, è molto meglio eseguire previsioni settimanali sul venduto dell'intera settimana, rispetto che giornaliere;
- Essendo una rete neurale, le elaborazioni richiedono una quantità consistente di tempo e risorse computazionali;
- Il software è proprietario (*closed source*) e prevede una sottoscrizione a pagamento.

Non è prevista la dismissione del modulo software, ma la sezione IT del gruppo vorrebbe comprendere quali sono le possibili soluzioni alternative a Microsoft Neural Network esistenti. Il percorso di tesi proposto dall'azienda ha avuto come oggetto l'arricchimento e l'elaborazione dei dati disponibili per poter studiare soluzioni alle problematiche descritte.

Poiché i diversi brand del gruppo hanno caratteristiche differenti, sia nei punti vendita che nei capi venduti, entrambi i progetti sono stati svolti limitandosi all'utilizzo dei dati del marchio Terranova, maggiormente presente all'interno del territorio italiano.

Dal punto di vista tecnico, l'unica richiesta è stata la compatibilità delle soluzioni proposte con i sistemi aziendali disponibili, in grado di eseguire codice in linguaggio R o Python.

Capitolo 2

Le tecniche utilizzate

All'interno di questo capitolo si descrivono le principali tecniche di clustering e previsione di serie storiche e i vari aspetti legati a questi due ambiti. Verrà fornito un background tecnico necessario a comprendere il funzionamento dei principali algoritmi disponibili e verranno descritti nel dettaglio tutti gli strumenti utilizzati.

2.1 Clustering

Con il termine *clustering* si indica un insieme di approcci in grado di individuare raggruppamenti di elementi all'interno di uno spazio multidimensionale, ed eventualmente definirne a posteriori la classe reale di appartenenza. L'obiettivo di questa disciplina è far si che gli elementi all'interno di un cluster siano *più simili* tra loro rispetto agli elementi degli altri cluster. L'attenzione a questa famiglia di approcci è in costante crescita anno dopo anno, in quanto il valore informativo che può generare una corretta clusterizzazione è estremamente importante. Alcuni dei principali ambiti in cui questo tipo di analisi risulta importante sono:

- Computer vision;
- Pattern recognition;
- Machine Learning;
- Bioinformatica.

Questo tipo di analisi viene definita **non supervisionata**, in quanto l'effettiva classe di appartenenza degli elementi da elaborare è sconosciuta, ed è quindi necessario effettuare ipotesi sulle caratteristiche degli elementi. Questo

comporta una **complessità** molto elevata per poter convergere a una soluzione ottima: ipotizzando di conoscere il numero di cluster n, per un qualunque dataset con un numero di elementi pari ad e, il numero di differenti soluzioni è dell'ordine di $\frac{n^e}{n!}$, ovvero un'approssimazione del numero di Stirling di seconda specie. Ma poiché, come generalmente avviene, è molto raro conoscere a priori il numero di cluster, anche questo valore deve essere determinato in qualche modo e il numero di possibili soluzioni aumenta vertiginosamente, poiché risulta necessario calcolare il valore precedente **per ogni valore da 1 a** n. Per questi motivi, il problema è considerato di tipo **NP Hard**.

2.1.1 Differenze con la classificazione

Il clustering non è da confondere con i tipici algoritmi di *classificazione*, dove le classi degli elementi appartenenti al dataset usato per addestrare il modello sono definite a priori. In questo tipo di problemi, detti **supervisionati**, l'obiettivo è riconoscere correttamente la classe degli oggetti partendo dalle loro proprietà. Al contrario, nella clusterizzazione l'algoritmo utilizzato deve raggruppare in un numero non fissato di classi gli elementi di cui si conoscono gli attributi, secondo un principio di **similarità** tra elementi. In figura 2.1 è possibile comprendere la differenza tra un dataset fornito a un algoritmo di clustering (figura 2.1a) e lo stesso dataset utilizzato con un algoritmo di classificazione (figura 2.1b).

Dataset

	Altezza	Peso
ID 1	170 cm	65 kg
ID 2	185 cm	80 kg
ID 3	150 cm	45 kg
...		.....
ID N	160 cm	60 kg

(a) Dataset senza classe target.

Dataset

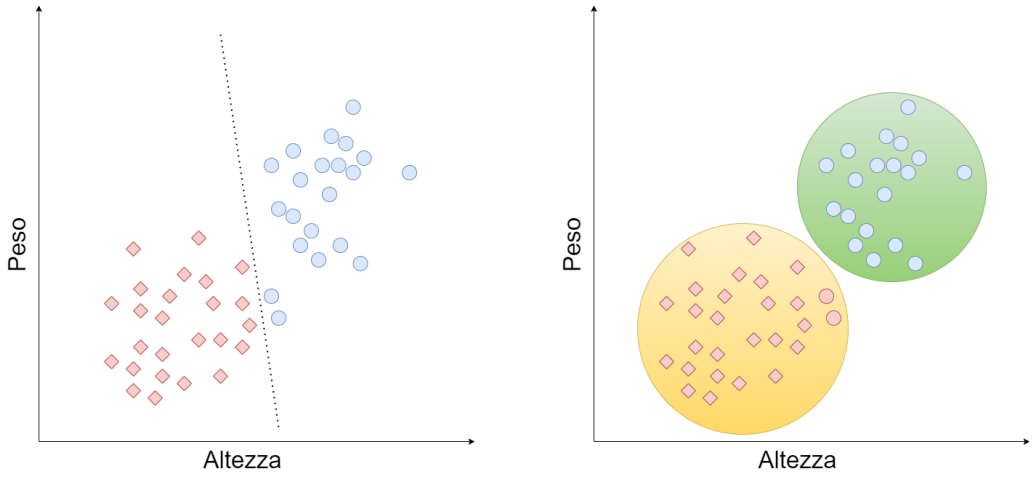
	Altezza	Peso	Classe
ID 1	170 cm	65 kg	Adulto
ID 2	185 cm	80 kg	Adulto
ID 3	150 cm	45 kg	Bambino
...		.....	
ID N	160 cm	60 kg	Bambino

(b) Dataset con classe target.

Figura 2.1: Dataset per apprendimento supervisionato e non.

Nello stesso modo, per i due approcci cambia anche l'obiettivo finale dell'addestramento:

- Se l'addestramento è di tipo supervisionato l'algoritmo deve cercare di separare nel migliore modo possibile i vari elementi appartenenti al dataset, riducendo il numero di classificazioni scorrette, come in figura 2.2a;
- Se l'addestramento non è supervisionato, come nel caso del clustering, il modello deve sviluppare un numero arbitrario di gruppi a cui associare gli elementi del dataset più simili tra loro, (risultato rappresentato in figura 2.2b).



(a) Risultato di una classificazione. (b) Risultato di un clustering.

Figura 2.2: Differenti risultati raggiungibili tramite tecniche di data mining.

Inoltre, al contrario della classificazione dove le classi sono definite a priori, in nessun algoritmo di clustering è garantito che il numero di cluster calcolato sia realmente corretto, poiché come verrà descritto all'interno della sezione 2.1.5 relativa alla validazione dei risultati, non esistono tecniche in grado di rispondere a questo tipo di domanda in modo diretto.

2.1.2 Tipi di clustering

Gli algoritmi di clustering si suddividono in due categorie distinte:

- Algoritmi **agglomerativi**;

- Algoritmi **divisivi**.

La sostanziale differenza è data dal modo che hanno per generare i cluster: l'agglomerativo, detto bottom-up, è un tipo di algoritmo che considera inizialmente ogni elemento un cluster a sé stante, per poi raggrupparli iterativamente fino al raggiungimento di particolari condizioni. Il tipo di clustering divisivo, detto top-down, parte da un unico gruppo contenente tutti gli elementi e passo dopo passo suddivide in sotto cluster.

Esistono poi altre caratteristiche che contraddistinguono un algoritmo di clustering, in base al funzionamento e al modo in cui vengono gestiti gli elementi:

- **Esclusività**: nel caso di clustering esclusivo, i punti del dataset devono appartenere solo a un cluster, mentre nel clustering non esclusivo esiste la possibilità che un elemento appartenga a più gruppi contemporaneamente;
- **Probabilità** di appartenenza a un cluster: detto anche fuzzy clustering, è l'opposto del clustering esclusivo, poiché ogni elemento appartiene a tutti i cluster con una probabilità che varia tra lo 0% e il 100% e la somma delle probabilità per ciascun punto deve essere 100%;
- **Parzialità**: se il clustering è parziale i punti del dataset potrebbero non appartenere a nessun cluster, nel caso opposto il clustering è detto *completo*;
- **Eterogeneità**: i cluster eterogenei possono avere densità e forme molto diverse tra loro, al contrario in cluster *omogenei* vengono predilette forme e densità simili.

Ognuna di queste caratteristiche definisce il risultato che l'algoritmo di clustering fornirà in output, il tipo di cluster e la forma che questi avranno: è quindi molto importante scegliere la giusta configurazione per i dati a disposizione e per il problema che si vuole cercare di analizzare.

2.1.3 Il concetto di similarità

Uno degli aspetti principali degli algoritmi di clustering è il metodo utilizzato per calcolare la *similarità* tra gli elementi. Anche questo tipo di scelta influisce in maniera diretta sul risultato della clusterizzazione, sulla forma, densità e dimensione di ogni singolo cluster. Il concetto di similarità si concretizza in una **metrica** (o funzione di distanza) che viene utilizzata dall'algoritmo per calcolare quanto siano lontani gli elementi forniti all'interno dello spazio multidimensionale. Ogni attribuito è una dimensione dello spazio (ad esempio bidimensionale nella figura 2.2b) e la metrica impiegata ha il compito di sintetizzare la distanza degli elementi, uno dall'altro.

Metrica

Per poter essere definita tale, una metrica d deve soddisfare tre assiomi:

1. **Identità**: $d(x, y) = 0 \Leftrightarrow x = y$
2. **Simmetria**: $d(x, y) = d(y, x)$
3. **Disuguaglianza triangolare**: $d(x, y) \leq d(x, z) + d(z, y)$

Esistono diversi metodi per calcolare la distanza tra due elementi, ognuna con il compito di favorire determinati tipi di cluster rispetto ad altri, le principali utilizzate sono:

- Distanza euclidea;
- Distanza di Hamming;
- Distanza di Mahalanobis;
- Distanza di Jaccard;
- Distanza di Manhattan;
- Distanza coseno;
- Distanza di Minkowski.

Non verranno descritte tutte nel dettaglio, sia perché non è questo l'obiettivo della tesi, ma soprattutto poiché non prettamente utili ai fini del progetto descritto. Infatti, alcune delle distanze elencate trovano utilizzo principalmente in altri ambiti (computer vision, text mining e semantic analysis).

Distanza euclidea

La distanza più semplice da comprendere: la distanza tra due punti p e q è la lunghezza del segmento avente come estremi i punti in questione. Formalmente, ipotizzando uno spazio euclideo in n dimensioni:

$$dist_{euclidea}(p, q) = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

È universalmente molto utilizzata nel clustering perché si adatta molto bene a qualunque esigenza, anche se non è esente da difetti:

- Nel calcolo della distanza le dimensioni più grandi delle altre hanno un peso specifico molto più alto, e questo rischia di compromettere il corretto risultato della clusterizzazione;

- Se il numero di dimensioni è troppo elevato faticherà a trovare elementi veramente simili tra loro.

Distanza di Manhattan

Detta anche *taxicab* o *cityblock*, è una semplificazione della distanza euclidea (la differenza è rappresentata in figura 2.3), in quanto è la somma delle differenze assolute per ogni dimensione dello spazio multidimensionale:

$$dist_{manhattan}(p, q) = \sum_{k=1}^{n} |p_k - q_k|$$

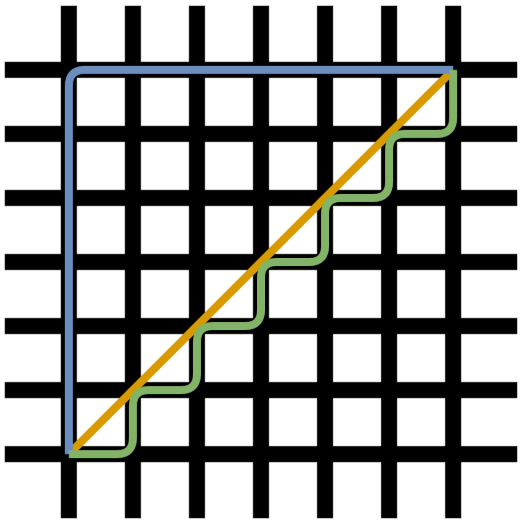


Figura 2.3: Differenza tra distanza euclidea (gialla) e Manhattan (blu e verde).

Distanza di Mahalanobis

Questa distanza è più complessa da calcolare rispetto alle precedenti, poiché si basa sul concetto statistico delle correlazioni tra variabili. Introdotta da Prasanta Chandra Mahalanobis nel 1936 in [Mah36], questa metrica è spesso usata al posto dell'euclidea in quanto tiene conto delle correlazioni dei dati e pesa le diverse componenti, considerando gli spazi di variazione e di correlazione.

Per calcolare la distanza di un punto nello spazio multidimensionale a grandezza n è necessario ottenere il valore medio μ per ogni dimensione in n, calcolando poi la matrice di covarianza S.

$$\mu = \frac{1}{n}\sum_{k=1}^{n} p_i \qquad S = \frac{1}{n}\sum_{i=1}^{n}(p_i - \mu)(p_i - \mu)^t$$

$$dist_m(p) = \sqrt{(p-\mu)^t \ S^{-1} \ (p-\mu)}$$

È possibile utilizzarla anche per la distanza tra due punti p e q con la stessa distribuzione e matrice di covarianza, tramite la formula:

$$dist_{mahalanobis}(p,q) = \sqrt{(p-q)^t \ S^{-1} \ (p-q)}$$

Distanza di Minkowski

La distanza di Minkowski è una generalizzazione di quella euclidea e della distanza di Manhattan. Utilizza infatti un intero p come ordine:

$$dist_{minkowski}(x,y) = (\sum_{k=1}^{n} |x_k - y_k|^p)^{\frac{1}{p}}$$

Se $p = 1$ questa metrica equivale alla distanza di Manhattan, con $p = 2$ rappresenta la distanza euclidea. **Può essere considerata una metrica solo per valori di** $p \geq 1$, poiché con interi più piccoli non vale più la proprietà di distanza triangolare (assioma 3 nella descrizione di metrica).

Distanza di Gower

Fino a questo momento è sempre stato descritto uno spazio multidimensionale degli attributi di tipo *numerico* (altezza, peso, ecc), ma è molto comune dover lavorare con ulteriori tipi di attributi diversi: i **categorici**. È molto complesso gestire un attributo categorico, che può essere:

- **Ordinale**, se i valori consentono l'ordinamento degli oggetti in base al valore dell'attributo (come ad esempio un voto: *ottimo*, *distinto*, *buono*);
- **Nominale**, se non è possibile ordinarli per valore ma solo distinguerli (ad esempio i nomi delle regioni italiane).

Con attributi ordinali è semplice eseguire la conversione in numero, ma questa pratica risulta complessa quando si tratta di attributi categorici nominali.

In letteratura sono presenti diversi modi di gestire la conversione, alcune di queste descritte all'interno di [Roy19], ma non sono gli unici modi di risolvere il problema. Un approccio alternativo è la distanza di Gower, descritta inizialmente in [Gow71]:

- Per gli attributi quantitativi (numerici) si calcola la distanza di Manhattan normalizzata sul range di intervallo dei valori;
- Gli attributi categorici ordinali vengono ordinati e successivamente trattati come attributi quantitativi;
- Ogni attributo categorico nominale è convertito in k colonne binarie (con $k =$ *numero di diversi valori dell'attributo*) e su questi si utilizza il coefficiente di Dice per calcolarne la distanza.

Per calcolare il coefficiente di Dice tra due elementi si conta il numero di dimensioni dove entrambi gli elementi hanno valore *true* (TT), le dimensioni dove il primo elemento ha valore *true* e il secondo *false* (TF), le dimensioni con il primo elemento *false* e il secondo *true* (FT) e per concludere le colonne dove entrambi hanno valore *false* (FF). Il valore di distanza per gli attributi categorici nominali è calcolato come segue:

$$dist_{dice} = \frac{TF+FT}{2\times TT+(TF+FT)}$$

Avendo quindi il valore di similarità per ogni dimensione, si accorpano in un unico valore tramite media. In questo modo viene calcolata la matrice delle distanze di ogni elemento con ogni altro, con valori che possono risultare nel range $[0-1]$ (sulla diagonale sono presenti solo "0" indicativi che la distanza tra sè stessi sia ovviamente nulla).

Questo approccio permette di utilizzare gli attributi categorici in maniera diretta, senza scegliere un modo per convertirli in valori numerici, ma presenta un difetto: **non verifica il terzo assioma** per le metriche (3).

Per questo motivo, in alcune tipologie di algoritmi, descritti nella prossima sezione 2.1.4 questa distanza non può essere utilizzata: si richiede una metrica in input.

2.1.4 Algoritmi

Dopo aver descritto i fondamentali dell'analisi cluster, si procede descrivendo le principali tecniche disponibili. Per eseguire correttamente una clusterizzazione è molto importante scegliere l'algoritmo di clustering che più si adatta alle caratteristiche dei dati trattati. La scelta dell'algoritmo di clustering influenza profondamente anche la forma e le densità dei cluster in output, ed è importante

considerare questo aspetto durante tutta la fase di analisi. Verranno descritti nel dettaglio tre dei più importanti approcci alla clusterizzazione, impiegati nelle fasi successive del progetto.

K-means

L'algoritmo K-means è senza dubbio il più famoso e conosciuto per quello che riguarda la clusterizzazione. È estremamente semplice da implementare, non richiede calcoli complessi ma solo pochi parametri in input (oltre ai dati da clusterizzare). Il suo funzionamento è incentrato sul concetto di **centroide**, ovvero il punto nello spazio multidimensionale che corrisponde al punto medio del cluster. In figura 2.4 sono raffigurati due centroidi di altrettanti cluster.

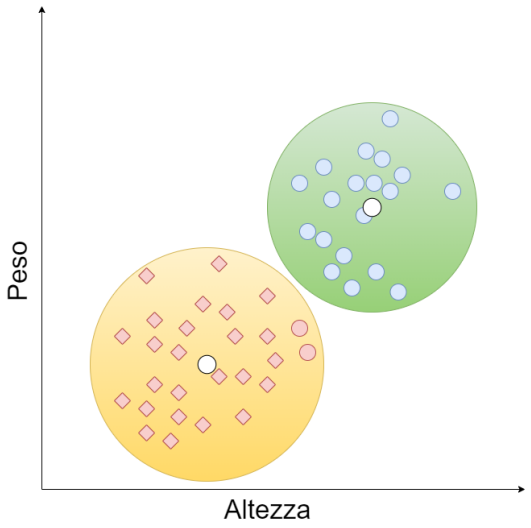


Figura 2.4: Centroidi (in bianco) di due cluster.

È importante notare che il centroide **non è un punto appartenente al dataset** passato in input all'algoritmo.

Per poter funzionare, K-means necessita anche di un valore k rappresentante il numero di cluster desiderati. Questo incarico è affidato ovviamente a chi esegue l'analisi, e non esiste una soluzione empirica da poter utilizzare a priori. In aggiunta a questo valore, richiede una **metrica** per calcolare la distanza dei punti dai vari centroidi. Per questo motivo, non è possibile utilizzare la distanza di Gower con K-means.

La dinamica di funzionamento di K-means è molto semplice: inizializza i centroidi iniziali con valori casuali all'interno dello spazio e a ogni iterazione associa gli elementi del dataset al centroide più vicino. Alla fine dell'iterazione ricalcola il centroide del cluster e riassocia ogni elemento al nuovo centroide

più vicino. Il processo si conclude quando da un'iterazione all'altra non ci sono modifiche o quando si raggiunge un certo numero di iterazioni.

Generalmente l'algoritmo è molto veloce ed efficiente e in un numero di iterazioni minore alla dimensione del cluster converge a una soluzione. K-means ha però tre difetti decisamente importanti:

- Non fornisce alcuna informazione sul **numero di cluster**;
- Si basa sul passaggio critico della **generazione iniziale dei centroidi casuale**;
- Genera cluster di forma globulare, ignorandone altri tipi o allungate.

Il primo problema è irrilevante se si conosce a priori il numero di cluster del dataset, ma anche nel caso in cui questo sia sconosciuto è comunque possibile effettuare ipotesi sul corretto numero di cluster **a posteriori**, eseguendo più prove con un valore k incrementale: questo argomento sarà oggetto di discussione nella prossima sottosezione 2.1.5.

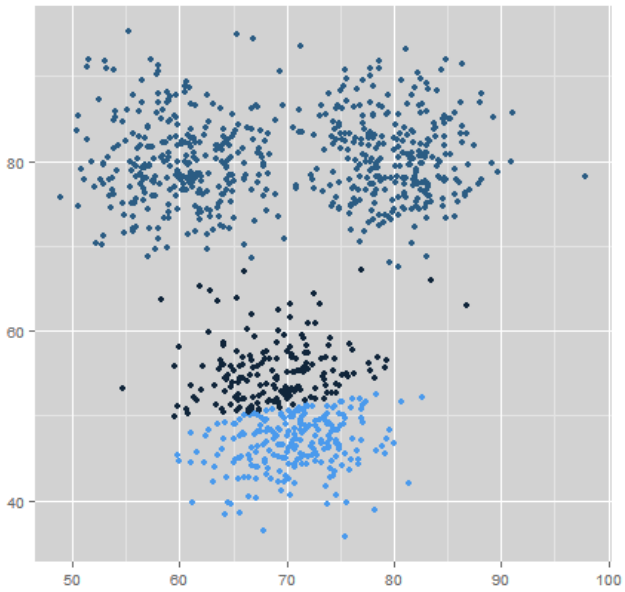


Figura 2.5: Output di K-means errato a causa dei centroidi iniziali, fonte dell'immagine: *edupristine*.

Per quanto riguarda la generazione iniziale dei centroidi, invece, il rischio è che a causa di una configurazione iniziale *particolare* l'algoritmo converga a una soluzione non adeguata, come in figura 2.5. Questo aspetto può essere affrontato in svariati modi, ad esempio eseguire più volte l'algoritmo per controllare se

effettivamente questo converga alla stessa soluzione, ma con dataset molto grandi la probabilità che questo sia sufficiente è molto bassa. Altri modi per evitare questa situazione sono:

- Campionare gli elementi del dataset;
- Utilizzare un algoritmo di clustering gerarchico per la scelta dei centroidi iniziali;
- Usare un numero di centroidi più alto e eliminare a posteriori quelli non ben separati dagli altri;
- Utilizzare una variante dell'algoritmo più sensibile a questo aspetto, come *K-means++* oppure *bisecting K-means*.

Per controllare la bontà della clusterizzazione svolta da K-means, si utilizza una misura di errore, generalmente lo **scarto quadratico medio**, calcolando la distanza tra i punti dei cluster identificati e il loro centroide. Non ci si può basare solo su questo aspetto: ipotizzando di utilizzare un valore k (numero di cluster) uguale alla quantità di elementi all'interno del dataset, ogni punto sarà un cluster a sé stante e lo scarto quadratico medio sarà nullo (ma la soluzione raggiunta avrà una valenza pressoché inutile). Per questi motivi generalmente si **penalizzano**, durante il calcolo dell'errore, soluzioni con un numero di cluster più alto, ed è molto importante durante l'analisi non fossilizzarsi solo sul valore dell'errore.

È possibile utilizzare il valore di errore per **stimare il numero di cluster**: si esegue K-means con un valore incrementale k, calcolando ogni volta lo scarto quadratico medio. Si genera il grafico che mette in relazione il valore k con l'errore generato, in un modo simile a quello rappresentato in figura 2.6. Nel punto in cui la curva **cambia pendenza**, è situato il numero di cluster ideale per i dati a disposizione.

DBSCAN

Acronimo di *Density-Based Spatial Clustering of Applications with Noise*, questo algoritmo è stato proposto in [Est+96] allo scopo di fornire un approccio completamente basato sulla **densità** dei punti, rispetto alla loro forma. L'obiettivo di questo approccio è collegare tra loro regioni di punti di sufficiente densità, ignorando in autonomia i punti considerati *outlier*, ovvero gli elementi appartenenti al dataset ma fuorvianti e anomali rispetto alla tendenza. Non necessita a priori di un **numero di cluster**, al contrario di K-means, poiché alla fine dell'elaborazione sarà in grado di indicarne il valore. Richiede invece due parametri in ingresso:

Figura 2.6: Utilizzo dello scarto quadratico medio per intuire il numero di cluster corretto.

- ϵ: ovvero la distanza massima da un elemento a cui possono trovarsi altri punti per poter essere definiti *vicini*;
- *minPts*: il numero minimo di vicini all'interno del raggio ϵ per poter considerare l'elemento parte attiva di un cluster.

Il funzionamento di questo algoritmo è semplice: ogni punto che conta un numero di vicini $\geq$ di *minPts* è considerato **core-point** (ovvero parte di un cluster), se il numero di vicini è inferiore ma comunque non nullo l'elemento viene etichettato come **border-point** (e comunque appartenente al cluster raggiungibile). Tutti gli altri punti vengono etichettati come **noise-point** e ignorati ai fini della clusterizzazione. In figura 2.7 è rappresentato il funzionamento dell'algoritmo DBSCAN con un valore $minPts = 4$.

Questo approccio facilita la creazione di cluster di forme diverse rispetto a quelle sferiche generate da K-means, anche **allungate**, ma l'intero algoritmo si basa quasi completamente sul concetto di densità: se nel dataset fornito in input esistono cluster naturali con diverse densità, solo quelli più densi verranno etichettati come cluster, mentre il resto sarà ignorato, come in figura 2.8.

Le prestazioni dell'algoritmo sono direttamente legate alla struttura dati che viene utilizzata per memorizzare le distanze tra i punti del dataset. L'operazione

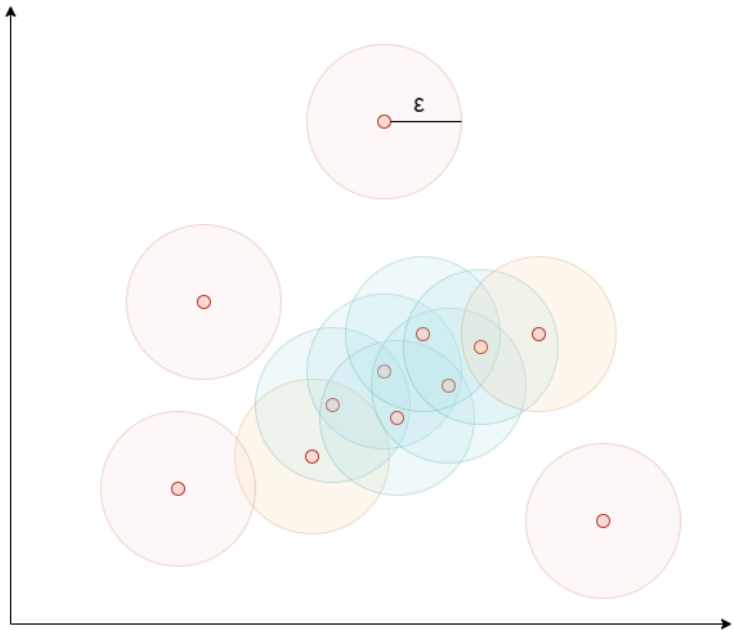


Figura 2.7: Esempio clusterizzazione DBSCAN, core-point in blu, border-point gialli e noise-point rossi.

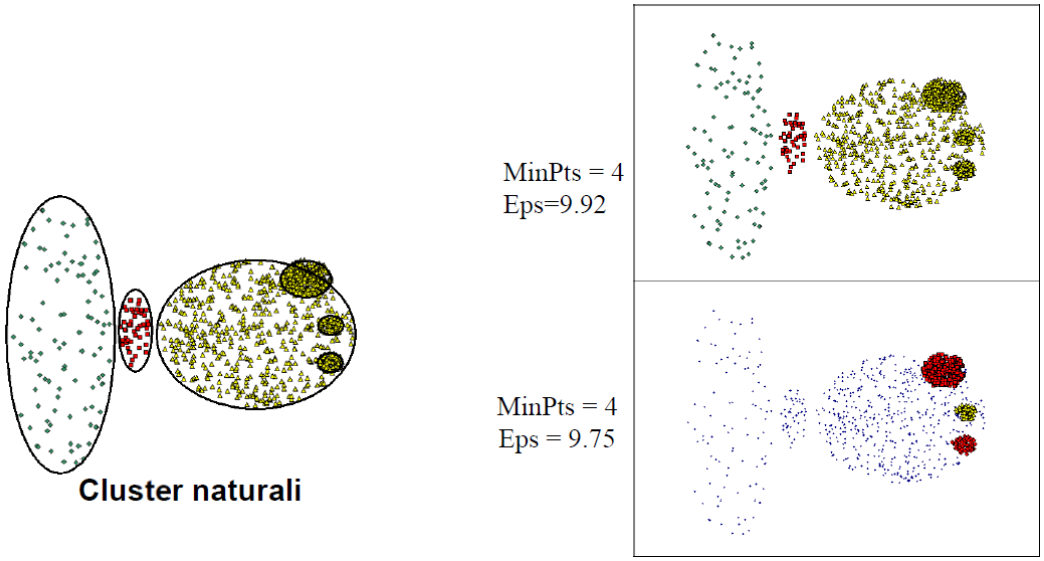


Figura 2.8: Clusterizzazione tramite DBSCAN di un dataset con densità variabili.

più costosa è ovviamente quella della ricerca dei vicini, ma se viene elaborata e memorizzata una matrice delle distanze che ha complessità esponenziale (cioè $O((n^2 - n)/2)$ perché basta la matrice triangolare superiore) DBSCAN è decisamente veloce.

La scelta dei parametri in input è delicata tanto quanto la scelta del numero di cluster in K-means: valori sbagliati possono portare a risultati deludenti anche con dataset facilmente clusterizzabili. Per scegliere ϵ si può utilizzare un grafico detto *k-distance-graph* (rappresentato in figura 2.9), dove sono rappresentate le distanze al k-esimo elemento (con $k = minPts$) ordinate in modo crescente, Nel punto in cui si verifica una *curva a gomito* nel grafico, quello è il valore di ϵ corretto.

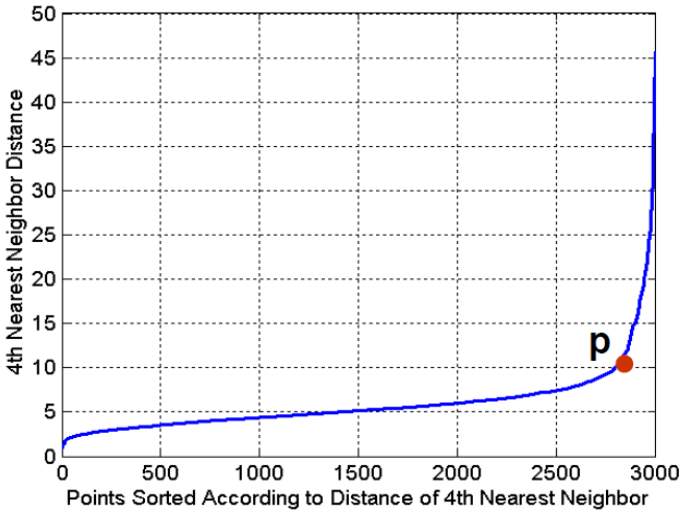


Figura 2.9: K-distance-graph per $k = 4$.

È importante tenere in considerazione che:

- Se ϵ è troppo piccolo la maggioranza degli elementi verrà scartata come noise-point;
- Se ϵ è troppo grande, verranno uniti tra loro troppi elementi, con la maggior parte in un unico gruppo.

Il valore di *minPts* viene generalmente scelto tenendo conto delle dimensioni del dataset: per elementi bidimensionali viene usato un valore $minPts = 4$, altrimenti per dataset con n attributi viene di norma scelto un valore $minPts = n + 1$.

Clusterizzazione Gerarchica

L'ultimo algoritmo di clustering che verrà descritto è quello gerarchico. Questo approccio è l'unico tra quelli analizzati ad accettare in input una matrice delle distanze al posto di una metrica (supportando implicitamente anche la distanza di Gower).

Alla fine dell'elaborazione l'algoritmo ritorna come output un dendrogramma (ovvero una **gerarchia di cluster**) rappresentato in figura 2.10. Il funzionamento di questo approccio è molto semplice: se l'algoritmo è di tipo **bottom-up** ogni elemento è un cluster a sé stante e a ogni iterazione i due cluster più simili tra loro vengono accorpati formandone uno nuovo. Si continua in questo modo finché non si raggiunge un unico cluster contenente tutti gli elementi. Nel clustering **top-down** al contrario si parte da una situazione opposta, dove tutti gli elementi appartengono a un unico cluster e via via vengono separati i due cluster più distanti.

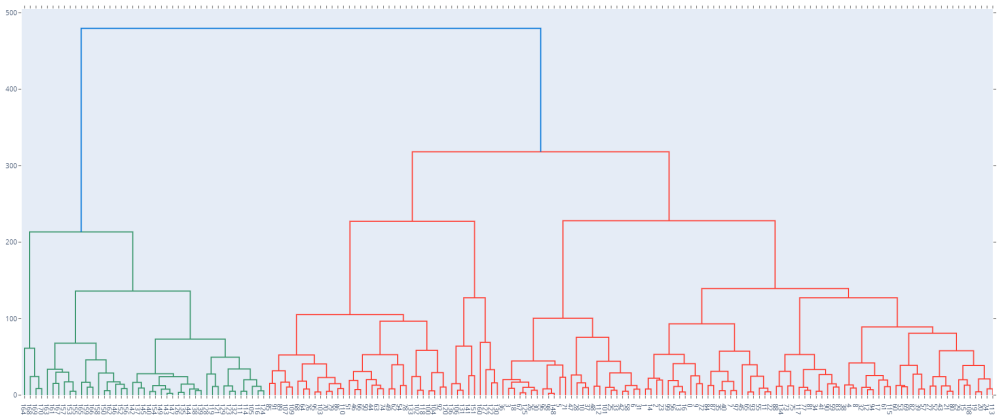


Figura 2.10: Esempio di dendrogramma generato dal clustering gerarchico.

Come anticipato, questo tipo di algoritmo non richiede per forza una metrica per funzionare, ma può essere eseguito anche su una matrice delle distanze precalcolata come quella generata dalla distanza di Gower.

Al contrario di K-means o DBSCAN, la clusterizzazione gerarchica richiede un metodo per calcolare la **distanza tra cluster**. Quando i cluster sono composti da un solo elemento ciascuno è semplice calcolare quanto disti uno dall'altro, quando un cluster contiene anche solo due elementi il calcolo risulta non più banale. Esistono diversi modi per calcolare la distanza tra cluster di dimensioni qualunque:

- Single Link (detto anche MIN);

- Complete Link (detto anche MAX);
- Group Average;
- Estraendo da ogni cluster il centroide e calcolando la distanza tra i due;
- Metodo di Ward.

MIN equivale a calcolare la minima distanza tra due elementi dei cluster, al contrario **MAX** restituisce la massima distanza calcolata nello stesso modo. Il metodo Single Link favorisce cluster non sferici, ma soffre di più gli outlier rispetto al metodo Complete Link, che a sua volta è limitato a cluster di forma globulare, e può capitare che suddivida anche cluster molto gradi.

Tramite il metodo **Group Average** si calcola la media di tutte le distanze generate dall'accoppiamento di tutti gli elementi di un cluster con tutti gli elementi dell'altro. È sensibilmente più complesso da calcolare rispetto a MIN o MAX, ma è una via di mezzo tra i due, e riesce a bilanciare pregi e difetti dei due metodi rimanendo però legato alle forme globulari.

Il metodo di **Ward** calcola quanto aumenta l'errore quadratico quando due cluster vengono fusi (per questo motivo richiede di calcolare il centroide per ogni cluster), e lo associa come valore di similarità tra i due. Anche in questo caso il principale difetto è la tendenza a generare cluster globulari, ma risulta più stabile al rumore e agli outlier degli altri. Può essere utilizzato per calcolare il numero di cluster k con cui inizializzare K-means.

Il principale svantaggio dell'approccio gerarchico è la **complessità computazionale**: per ottimizzare i calcoli la matrice delle distanze deve essere precalcolata e mantenuta in memoria, ma ad ogni ciclo deve poi essere aggiornata e vanno ricalcolate le nuove distanze con i cluster appena creati. Questo porta la complessità temporale a $O(n^3)$, rendendolo praticamente impossibile da utilizzare con dataset di grandi dimensioni.

2.1.5 Tecniche di validazione dei cluster

Dopo aver descritto i diversi modi di approcciare un problema di clustering, si analizzano i vari metodi di validazione dei risultati ottenuti in output. L'analisi clustering appartiene alla classe di problemi **non supervisionati**, quindi per definizione non è possibile calcolare quanto l'output dell'algoritmo impiegato sia distante dalla realtà. È possibile utilizzare tecniche di validazione legate al mondo **supervisionato** solo nel caso in cui siano disponibili le etichette reali degli elementi, ma è una situazione decisamente atipica. Poiché i dati disponibili per questo progetto di tesi (descritti nel capitolo 3) non presentano l'attributo classe, verranno descritte le due principali misure di validità **esterne**:

- Silhouette;
- Statistica di Hopkins.

Silhouette

Questa misura serve a quantificare quanto i cluster siano ben separati uno dall'altro (**separazione**) e contestualmente quanto i vari gruppi di elementi siano simili (**coesione**). Sia il valore di separazione che quello di coesione possono essere calcolati in due modi diversi:

1. Considerando i cluster come grafi, i cui nodi sono gli elementi e gli archi tutti i possibili collegamenti da un nodo all'altro;
2. Considerando il centroide del cluster.

Nel primo caso la coesione viene calcolata sommando tra loro il peso degli archi tra nodi appartenenti a un cluster, mentre per la separazione si considerano gli archi dei nodi appartenenti a cluster distinti. Nel caso in cui sia possibile calcolare il centroide del cluster, la coesione si calcola sommando le distanze dai singoli elementi al centroide, mentre per la separazione si utilizza la distanza tra i centroidi. Un riepilogo grafico di quello che avviene è disponibile in figura 2.11 per modelli basati su grafi e in figura 2.12 per i modelli basati centroidi.

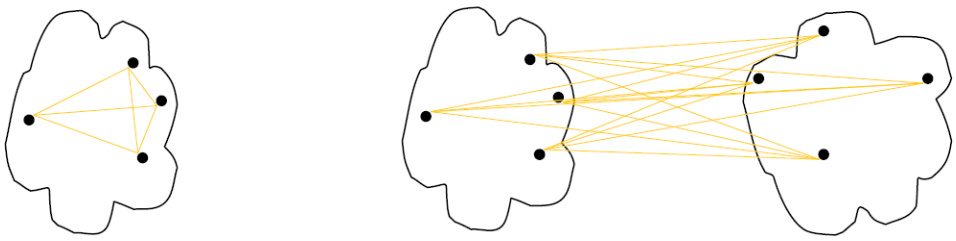


Figura 2.11: Coesione e separazione di modelli basati su grafi.

La silhouette è la misura che cerca di sintetizzare questi due elementi (coesione e separazione) all'interno di **un unico valore**. Viene calcolato (per ogni punto all'interno del dataset) un valore a_i che indica la **distanza media dell'elemento da tutti gli altri all'interno del proprio cluster** C_i. Successivamente viene calcolato il valore b_i, rappresentante **la più piccola distanza media del punto da tutti gli altri cluster**:

$$a_i = \frac{1}{|C_i| - 1} \sum_{j \in C_i} dist(i, j) \qquad b_i = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} dist(i, j)$$

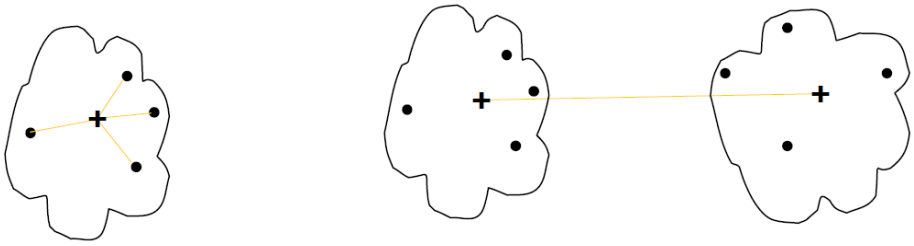


Figura 2.12: Coesione e separazione di modelli basati su centroidi.

Infine viene sviluppato, sempre per ogni punto, il proprio *silhouette score*, ovvero un punteggio che varia tra -1 e 1 secondo la formula:

$$silhouette(i) = \frac{(b_i - a_i)}{max(a_i, b_i)}$$

Punteggi **vicini a 1** sono preferibili (equivalenti a un valore a_i piccolo e b_i grande) in quanto implicano cluster coesi e ben separati dagli altri. Più ci si avvicina a 0 (ovvero b_i si avvicina ad a_i) e più l'elemento preso in questione è troppo vicino ad altri cluster. Nel caso peggiore in cui si abbia un risultato negativo il punto in questione è più vicino ai punti di cluster a cui non appartiene rispetto a quelli dello stesso cluster a cui è stato assegnato (quindi b_i è minore di a_i).

Se si esegue quest'analisi per ogni punto del dataset, è possibile generare un grafico simile a quello in figura 2.13, dove rappresentare i valori di silhouette per ogni punto. È inoltre possibile calcolare il valore di **silhouette medio** per la clusterizzazione (in figura 2.13 è la linea tratteggiata rossa nel grafico a sinistra), che potrà dare un indicazione di quanto effettivamente sia stata svolta correttamente.

Questo valore può essere utilizzato, ad esempio con K-means, per scegliere il corretto numero di cluster: si esegue K-means con un valore k incrementale, calcolando ad elaborazione conclusa il relativo valore di silhouette. Si potrà scegliere poi il valore k **associato a una silhouette migliore**, ignorando lo scarto quadratico medio che si riduce all'aumentare di k.

Statistica di Hopkins

Definito per la prima volta in [HS54], questo valore matematico può essere utilizzato per stimare quanto i dati contenuti nel dataset di partenza siano **distribuiti in modo uniforme**. Per stimare la propensione al clustering:

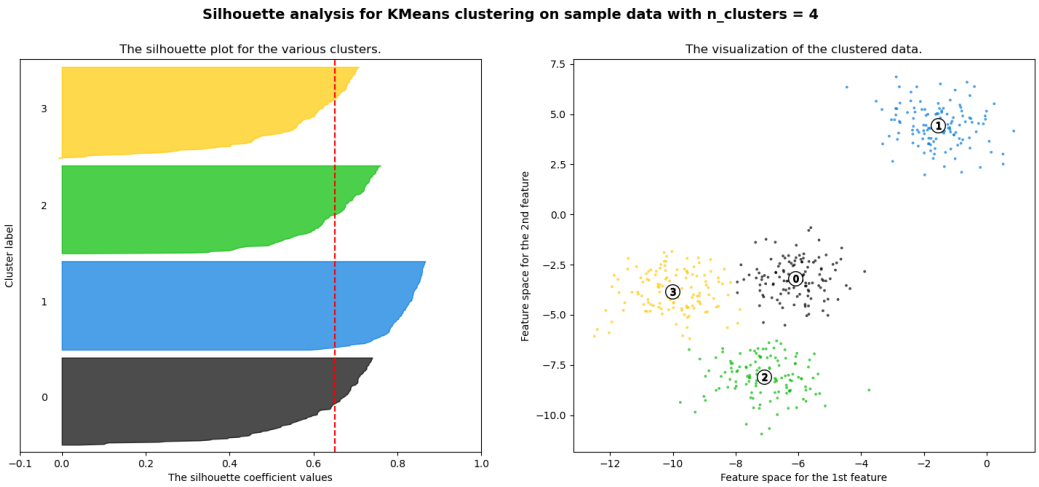


Figura 2.13: Esempio di analisi grafica silhouette, fonte immagine: *scikit-learn*.

1. Si genera un dataset parallelo D_{new} di m punti a quello fornito in input D_{real} (composto da n elementi, con $n >> m$), distribuendo gli elementi in maniera casuale nello stesso spazio multidimensionale;

2. Per ogni punto $x_i \in D_{new}$ si calcola u_i, la distanza al punto più vicino appartenente al dataset reale D_{real};

3. Si campionano altri m punti dal dataset reale $q_i \in D_{real}$ e per ognuno di essi si calcola la distanza w_i dal punto al suo vicino nel dataset reale D_{real}.

Infine si calcola la statistica di Hopkins tramite la seguente formula:

$$H = \frac{\sum_{i=1}^{m} u_i}{\sum_{i=1}^{m} u_i + \sum_{i=1}^{m} w_i}$$

Se si ottiene un H tra 0.3 a 0.7 il dataset è scarsamente clusterizzabile (la distanza dei punti all'elemento più vicino è circa la stessa sia per i punti generati sia per quelli campionati). Con un H vicino a 1 i dati sono ben clusterizzabili (valori di u_i molto piccoli), mentre per H tendenti a 0 i dati sono distribuiti in modo regolare e scarsamente clusterizzabili (valori di w_i molto piccoli).

Questo approccio **soffre le dimensionalità elevate** dei dataset, ed è quindi applicabile prevalentemente a elementi con un ridotto numero di attributi e, in genere, solo a spazi euclidei. Benché questo metodo possa fornire un'indicazione generica della direzione che si sta percorrendo con il dataset elaborato, rimane comunque un approccio basato su ipotesi e regole empiriche

i cui risultati sono da interpretare durante la fase di analisi, senza affidarsi ciecamente al giudizio di questa statistica.

2.2 Previsione storica

Dopo aver descritto nel dettaglio tutti gli aspetti relativi alla clusterizzazione, si passa al secondo obiettivo di questa tesi: l'analisi di una serie storica per poter stimarne l'andamento futuro. Una serie storica è un insieme di dati osservati su uno specifico fenomeno (ad esempio le vendite di un negozio) ordinati secondo la variabile temporale. Quest'ultima definisce la cadenza temporale in cui viene effettuata ogni osservazione: oraria, giornaliera, settimanale, trimestrale, annuale, etc (in figura 2.14 ad esempio ogni punto sul grafico corrisponde a una diversa giornata).

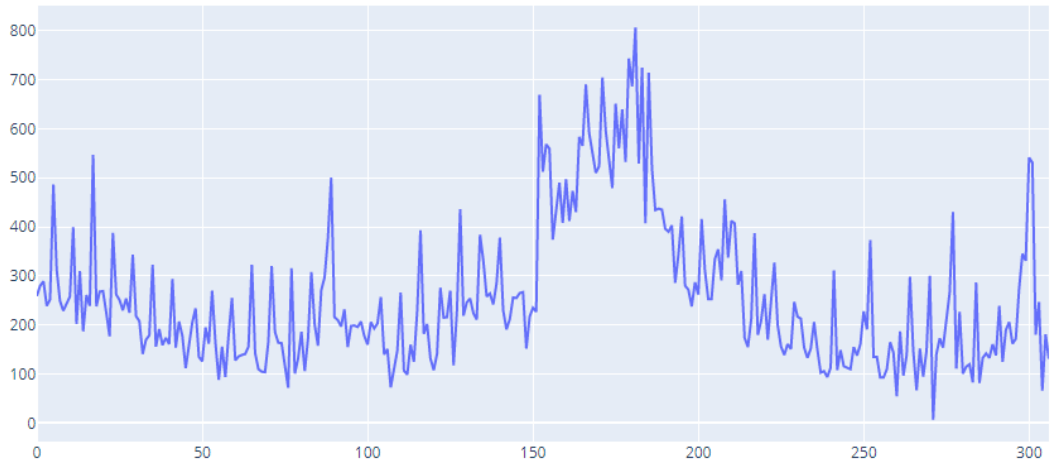


Figura 2.14: Una serie storica giornaliera.

Questo tipo di analisi è molto importante a livello aziendale, perché permette di organizzare diverse fasi critiche (dalla logistica alla produzione) di un futuro prossimo, basandosi sui soli dati disponibili nel presente, ad esempio:

- Stimare la domanda di un prodotto e programmarne la produzione in modo da soddisfare la richiesta;
- Ipotizzare la quantità di passeggeri che utilizzeranno una linea metropolitana e adeguare il personale di sicurezza incaricato di gestire la stazione;
- Prevedere la richiesta di forza lavoro per la produzione di un particolare bene.

Le previsioni possono esser svolte su periodi brevi, come i giorni o le settimane successive, ma anche a lungo termine (ad esempio quando è necessario stimare trend di crescita sul lungo periodo). Di solito, in output (oltre alle previsioni) viene indicato anche l'**intervallo di affidabilità**, che specifica quanto le stime siano *sicure* (rappresentato in figura 2.15). Ovviamente, progredendo con la richiesta di previsione l'intervallo di affidabilità sarà più ampio, in quanto l'accuratezza dell'algoritmo cala al progredire della stima.

Figura 2.15: Esempio di intervallo di affidabilità.

2.2.1 Componenti di una serie

Una serie storica può presentare, generalmente, alcuni andamenti e oscillazioni che prendono il nome di **componenti**. Questi elementi sono molto utili e devono essere riconosciuti in fase di analisi per poter essere gestiti e utilizzati nella previsione, e sono:

- **Trend**: detto anche *tendenza*, è presente in serie temporali che subiscono una crescita o diminuzione nel lungo termine dei valori, generalmente con un tasso di variazione costante;
- **Ciclo**: la componente ciclica è un pattern nella serie storica che si ripete nel tempo con cadenza più o meno regolare. L'oscillazione può presentarsi in condizioni specifiche, come espansioni o contrazioni economiche del fenomeno preso in esame;
- **Stagionalità**: oscillazioni generate principalmente da fattori climatici come l'alternanza delle stagioni, si riconosce di solito in pattern ripetitivi individuabili su periodi (annuali, mensili, settimanali, etc);
- **Accidentalità**: componente che si sviluppa da movimenti irregolari, ovvero eventi casuali che non sono predicibili;

- **Occasionalità**: una componente dovuta a eventi particolari, come innovazioni tecnologiche, crisi politiche o eventi bellici (o una pandemia globale con conseguente crisi sanitaria). Se l'effetto sulla serie si smorza velocemente il trend non subisce variazioni particolari, se al contrario la tendenza subisce un una variazione perenne (grafico discontinuo) è necessario suddividere la serie in due parti e analizzarle singolarmente.

La cosa più semplice da identificare, anche ad occhio, è il trend della serie. In figura 2.16 sono rappresentate due serie con e senza trend: nel primo caso viene detta **stazionaria**, mentre la seconda **non stazionaria** o **evolutiva**.

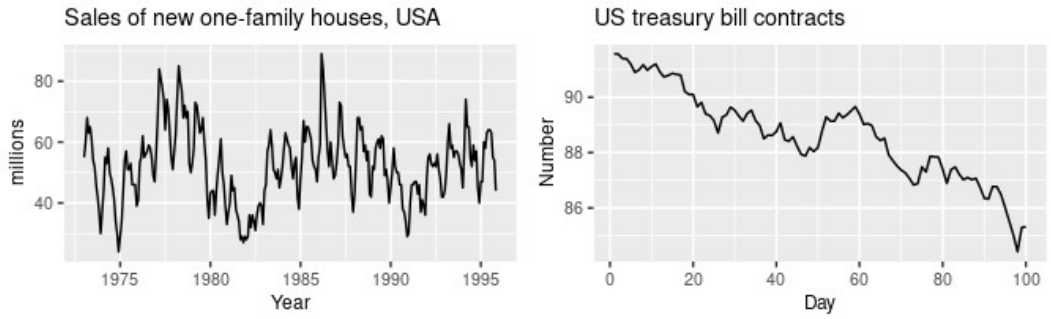


Figura 2.16: Esempi di serie storiche: stazionaria a sinistra ed evolutiva a destra, fonte dei grafici: [HA18].

Per riconoscere oscillazioni stagionali viene generalmente utilizzato un *seasonal plot*, un grafico dove vengono sovrapposti i dati relativi alle stesse stagionalità. Si rappresenta il solo periodo sull'asse delle ascisse, come in figura 2.17. In questo modo si possono *appaiare* più serie storiche appartenenti a periodi diversi, così da poter controllare a colpo d'occhio se esiste o no una componente stagionale.

Per rimuovere un trend a una serie storica si utilizza un procedimento chiamato **differenziazione**, sottraendo ai valori correnti della serie i valori assunti al tempo precedente (il risultato che si ottiene è rappresentato in figura 2.18). Una differenziazione del primo ordine y'_t (utile quando il trend è **lineare**) si ottiene nel seguente modo:

$$y'_t = y_t - y_{t-1}$$

Ovviamente se il trend è più complesso servirà una differenziazione di ordine più alto (secondo, terzo e così via).

È possibile rimuovere tramite differenziazione anche la componente stagionale, sottraendo al dato corrispondente il valore della stagione precedente:

$$y'_t = y_t - y_{t-m}$$

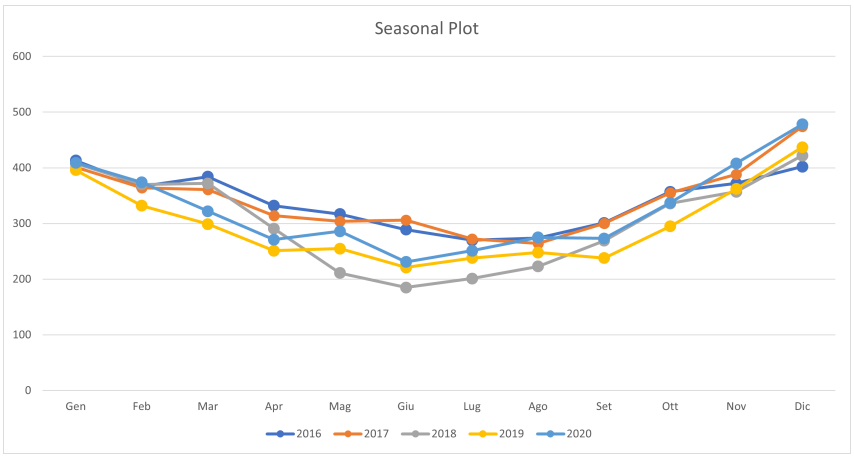


Figura 2.17: Seasonal plot su serie storica del consumo energetico domestico.

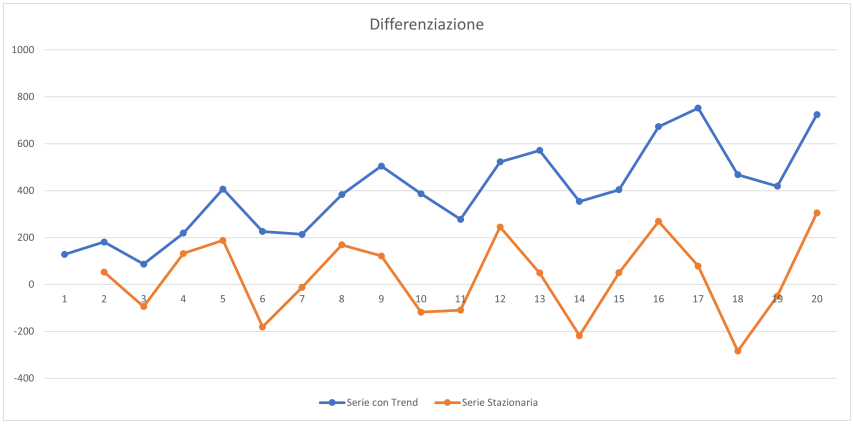


Figura 2.18: Differenziazione lineare di una serie evolutiva in stazionaria.

2.2.2 Definizione formale

I concetti definiti nella sottosezione precedente possono essere formalizzati in termini logici nel seguente modo:

- Per il trend di lungo periodo viene usata la variabile μ_t;
- Per l'andamento stagionale di periodo L si utilizza s_t;
- Eventi ciclici si rappresentano con c_t;
- Il residuo casuale e imprevedibile si riassume con r_t.

Queste componenti si possono combinare in diversi modi, le tre principali sono:

- **Modelli additivi**: $y_t = \mu_t + S_t + C_t + r_t$;
- **Modelli moltiplicativi**: $y_t = \mu_t \times S_t \times C_t \times r_t$;
- **Modelli misti**: $y_t = \mu_t \times S_t \times C_t + r_t$;

2.2.3 Funzionamento della previsione

Dopo aver definito le caratteristiche tipiche delle serie storiche, si approfondiscono i metodi di previsione, le varie tipologie di algoritmo disponibili e in che modo è possibile eseguire la validazione dei risultati ottenuti.

Come con il clustering, anche per la previsione storica esistono delle misure di bontà delle stime. Al contrario della clusterizzazione, però, la previsione di una serie è un problema **supervisionato**: si è al corrente del valore della serie su un intervallo temporale preciso, e dopo aver addestrato un modello utilizzando i dati precedenti all'intervallo *target* si confrontano i risultati. Ipotizzando di utilizzare un intervallo di tempo $[1, \ldots, n]$ per il quale sono disponibili i dati osservati $X = (x_1, \ldots, x_n)$, dopo aver generato le previsioni $F = (f_1, \ldots, f_n)$ è possibile calcolare l'accuratezza del modello nei seguenti modi:

- **BIAS** - media aritmetica degli errori: $= \frac{\sum_{i=1}^{n}(x_i - f_i)}{n}$
- **MAD** - media degli errori assoluti: $= \frac{\sum_{i=1}^{n}|x_i - f_i|}{n}$
- **MSE** - media degli errori quadratici: $= \frac{\sum_{i=1}^{n}(x_i - f_i)^2}{n}$

- **SE** - l'errore standard: $= \sqrt{MSE} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - f_i)^2}{n}}$
- **MAPE** - media degli errori assoluti in percentuale: $= \frac{\sum_{i=1}^{n} \frac{|x_i - f_i|}{f_i} * 100}{n}$

La più utilizzata è generalmente la misura di **errore standard** (a volte indicata anche come RMSE, *root mean squared error*) e verrà presa come riferimento per il calcolo dell'errore all'interno di questa tesi.

2.2.4 Modelli autoregressivi

La principale famiglia di approcci per il forecast di una serie storica è quella dei modelli autoregressivi a media mobile, detta anche **ARMA**. Spesso chiamata anche modello di Box-Jenkins [Box+15] dal nome dei suoi inventori, è un tipo di modello matematico lineare utilizzato per lo studio delle serie storiche dei dati.

Per serie stazionarie è possibile utilizzare un **modello autoregressivo** di ordine p, detti $AR(p)$. Per eseguire la previsione si somma una componente casuale, una componente costante e una componente lineare delle ultime p osservazioni secondo la formula:

$$y_t = cost + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t$$

Dove y_t è il valore della serie al tempo t della previsione, ϵ_t è la componente casuale e $\phi_{1,\dots p}$ sono i parametri del modello passati. Maggiore sarà il valore p, maggiore sarà il numero di valori precedenti analizzati. Per serie stazionarie semplici i risultati di questo modello sono molto soddisfacenti.

Un altro metodo per eseguire previsioni è l'utilizzo di **modelli a media mobile** di ordine q (detti $MA(q)$), dove il valore futuro di una variabile è ipotizzato essere uguale alla media delle osservazioni più una combinazione lineare degli ultimi q errori. Il tutto è riassumibile nella formula:

$$y_t = \mu + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t$$

Dove μ è il la media della serie e $\theta_{1,\dots q}$ sono i parametri del modello.

Se si combinano entrambe le componenti AR e MA si ottiene un modello di classe più generale: $ARMA(p, q)$. Questi modelli aggregano entrambi gli approcci analizzati, secondo la formula:

$$y_t = cost + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t$$

In questo modo, la media mobile (MA) corregge le previsioni del modello autoregressivo (AR), ma è un approccio utilizzabile solo con serie storiche stazionarie, senza trend o stagionalità.

Per approcciare questo tipo di serie è necessario passare a modelli **autoregressivi integrati a media mobile** (detti anche $ARIMA(p, d, q)$). Questo approccio lavora su una serie storica differenziata, utilizzando l'ordine d come valore per la differenziazione della serie (eseguire $ARIMA$ con un ordine $d = 0$ equivale ad utilizzare un modello $ARMA$).

La stagionalità non viene comunque gestita, ed è per questo che in [Box+15] è stato proposto un modello applicabile anche a dati che presentano la componente stagionale. Viene eseguita una **differenziazione stagionale** prima di eseguire la previsione, che richiede, oltre ai classici parametri (p, d, q) necessari per i periodi, anche un secondo set (P, D, Q) da applicare alla stagionalità. È inoltre richiesto il numero di periodi contenuti all'interno di una stagione: il parametro m. Se ad esempio la serie in questione presenta una stagionalità trimestrale, è necessario indicare un valore $m = 3$, se la stagionalità è annuale $m = 12$. Questo modello prende il nome di $SARIMA(p, d, q)(P, D, Q)m$.

Tra quelli descritti, questo è il modello più complesso ma in grado di adattarsi a una molteplicità di serie storiche diverse. Sfortunatamente, il numero di parametri richiesto può influire negativamente sulle performance della previsione poiché la scelta dei parametri in input non è banale.

Per poter facilitare l'analisi dei dati è stato sviluppato nel tempo un algoritmo in grado di eseguire una ricerca euristica sulla configurazione migliore per la serie storica analizzata: **auto ARIMA**.

Viene impostato per **ciascun parametro** un **intervallo di valori**, si esegue $SARIMA$ con una specifica combinazione di questi valori e successivamente si calcola l'errore della previsione. L'obiettivo di auto ARIMA è la ricerca del modello con le migliori performance, in grado di ridurre l'errore al minimo. L'algoritmo può funzionare in due modi diversi:

- In modo **euristico**: effettua una discesa del gradiente sul valore di errore, cercando di minimizzarlo fino a trovare la configurazione di parametri migliore;
- In modo **esaustivo**: tramite una *grid-search* che esegue tutte le possibili configurazioni generabili dagli intervalli forniti.

La prima è ovviamente molto più veloce ma eseguendo una ricerca euristica può convergere a un punto di minimo relativo. La *grid-search*, al contrario, trova il valore minimo assoluto, al prezzo di eseguire tutte le possibili configurazioni. Benché quest'ultima soluzione fornisca più garanzie, è generalmente utilizzata la ricerca euristica, in quanto raggiunge comunque buoni risultati, che differiscono

veramente di poco dall'ottimo assoluto, ma in una frazione di tempo rispetto alla *grid-search*.

2.2.5 Prophet

Un metodo alternativo alla famiglia dei modelli SARIMA è stato implementato nel 2018 dal team Data Science di Facebook, proposto in [TL18]. La procedura chiamata *Prophet* si basa su un modello additivo con supporto a trend non-lineari e stagionalità annuale, settimanale e giornaliera. Supporta la possibilità di integrare nel modello anche le festività nazionali, in aggiunta a qualunque tipo di **variabile esogena**. I principali pregi dell'algoritmo sono:

- Supporto anche a serie storiche con dati mancanti, senza danneggiare il trend o le previsioni;
- Gestione integrata degli outlier;
- Completamente **automatico**, non richiede parametri;
- Costo computazionale di creazione e addestramento del modello estremamente basso.

In figura 2.19 è rappresentata la scomposizione effettuata da Prophet su un dataset di prova, considerando una stagionalità annuale e contestualmente una settimanale.

2.2.6 Approcci neurali

Le serie storiche possono essere analizzate e processate anche tramite modelli basati su algoritmi **deep learning**. Presentano una maggiore complessità computazionale e richiedono più dati per poter formulare previsioni precise, ma generalmente ottengono risultati migliori dei semplici modelli autoregressivi. Spesso sono inoltre associati a servizi aggiuntivi (ad esempio il supporto al meteo per Microsoft Neural Network) che permettono di migliorarne le prestazioni.

Il più grande difetto che possiedono è la loro struttura a *black-box*: non viene fornita motivazione alla previsione data. Questo aspetto può essere trascurabile sotto il profilo pratico, perché spesso in molte realtà aziendali *l'importante è che funzioni*, ma in certi ambiti sapere perché una previsione ha dato un certo esito può essere molto importante.

Questi approcci non verranno descritti o utilizzati all'interno di questa tesi, ma per completezza viene citato **Amazon Forecast**. Questo algoritmo proprietario presenta numerosi vantaggi:

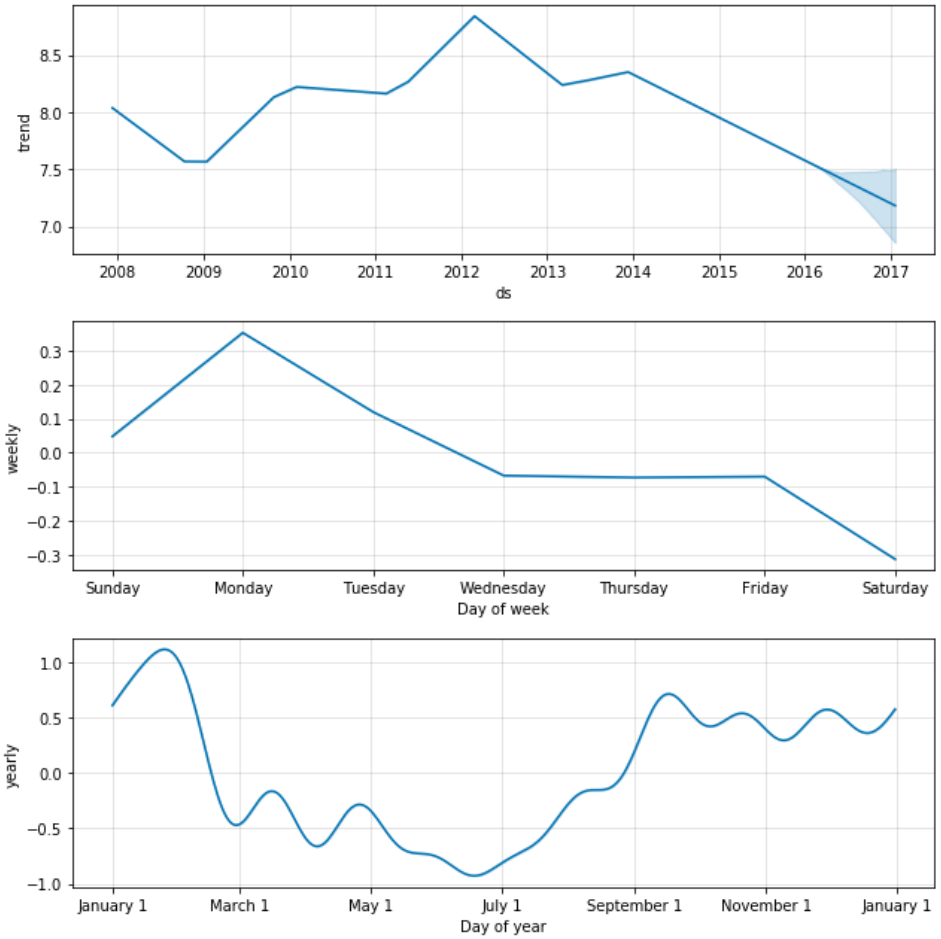


Figura 2.19: Esempio di scomposizione tramite Prophet.

- Non richiede competenze informatiche particolari per poter essere utilizzato;
- Un singolo modello viene addestrato su numerose serie storiche diverse (non solo una), e può generare previsioni per ognuna di esse;
- Autogestito sui server Amazon Web Services, non richiede particolari configurazioni lato infrastrutturale;
- Supporto al *cold start*, ovvero la possibilità di generare previsioni anche per serie storiche non particolarmente popolate utilizzando altre serie *simili*;
- Possibilità di automatizzare la raccolta dei dati, la generazione delle previsioni e la visualizzazione dei risultati.

Ovviamente tutto questo ha un costo: mantenere i dati sui server Amazon (poiché è utilizzabile soltanto sulle infrastrutture proprietarie AWS), il tempo di computazione necessario ad addestrare il modello e il costo **per ogni singolo forecast di ciascuna serie storica**.

Capitolo 3

I dati aziendali

Dopo aver definito i vari strumenti utilizzabili per clustering e forecast, si descrivono i dati forniti dall'azienda per svolgere entrambe le fasi: sono in tutto sei file relativi a diversi aspetti dei punti vendita:

- L'anagrafica dei negozi contenente le informazioni di base dei punti vendita;
- Le tipologie di articoli venduti (chiamati *moduli*);
- La superficie d'esposizione occupata dai negozi;
- I dati relativi al flusso di clienti giornaliero;
- Lo storico giornaliero delle vendite per singolo negozio;
- Le previsioni meteo utilizzate da Microsoft Neural Network per eseguire il forecasting.

Per ognuno di essi verranno analizzati gli attributi presenti e verrà descritto a grandi linee in che modo è stato eseguito il preprocessing dei dati (secondo la metodologia CRISP-DM, definita da [She00]).

3.1 Anagrafica negozi

Il primo dataset analizzato è quello relativo alle informazioni dei punti vendita, contenente 211 righe. Ognuna di esse è associata agli attributi descritti in tabella 3.1.

In tabella 3.2 è sono rappresentate le prime otto righe relative ad altrettanti punti vendita prese dal dataset in questione.

Attributo	**Formato**	**Descrizione**
`unit_sk`	*intero*	Identificatore numerico del punto vendita, utilizzato come chiave nel database principale.
`region_cod`	*stringa (3)*	Codice della regione d'appartenenza.
`region_des`	*stringa*	Nome per esteso della regione d'appartenenza.
`province_cod`	*stringa (2)*	Codice della provincia di appartenenza.
`latitude`	*intero*	Il valore di latitudine del punto vendita.
`longitude`	*intero*	Il valore di longitudine del punto vendita.
`openingdate`	*intero*	La data di apertura del negozio.
`closingdate`	*intero*	La data di chiusura (eventuale) del negozio.

Tabella 3.1: Gli attributi del dataset anagrafica.

unit_sk	region_cod	region_des	province_cod	latitude	longitude	openingdate	closingdate
12584	SAR	SARDEGNA	OT	40.922.512	9.486.853	20091205	99991231
12894	PIE	PIEMONTE	TO	45.076.266	7.671.702	20100904	99991231
13428	PUG	PUGLIA	LE	40.142.393	18.492.168	20110627	99991231
14173	SAR	SARDEGNA	OG	39.926.406	9.661.878	20111126	99991231
14925	PIE	PIEMONTE	BI	45.562.903	8.058.178	20120721	99991231
15493	CAL	CALABRIA	CZ	38.910.535	16.587.757	20121117	99991231
16538	PUG	PUGLIA	LE	40.055.550	17.990.005	20140628	99991231

Tabella 3.2: Estratto delle prime righe di anagrafica punti vendita.

Durante la fase di analisi dei dati sono stati rimossi quindici punti vendita, associati a **coordinate nulle**, e sono state manipolate le **coordinate geografiche** (per formato testuale errato) e le **date di apertura e chiusura**, in modo da poterle utilizzare in maniera agevole.

Dopo aver analizzato nuovamente il dataset, è stato possibile rimuovere la colonna `closingdate` in quanto tutti i 196 negozi restanti sono ancora in attività (ovvero contengono un attributo uguale a `31-12-9999`). Prima di passare al dataset successivo sono state memorizzate le **coordinate** dei punti vendita in un dataset a parte, in quanto necessarie per il processo di arricchimento dei dati descritto nella sezione 3.6.

3.2 Moduli venduti

Il secondo dataset analizzato descrive i cosiddetti *moduli*, un termine aziendale utilizzato per indicare le categorie di prodotti vendute:

- *Donna* (modulo 01);
- *Uomo* (modulo 02);
- *Bimba* (modulo 10);
- *Bimbo* (modulo 11);
- *Intimo* (modulo 20).

Poiché un punto vendita può in qualunque momento ampliare la propria offerta commerciale, aggiungendo o sostituendo moduli presenti, in questo dataset sono disponibili anche due attributi di impiego e dismissione modulo. La struttura completa del dataset è disponibile il tabella 3.3.

Un estratto delle entry del dataset, relativo a 207 diversi punti vendita, è disponibile in tabella 3.4, dove è possibile notare che i moduli ancora venduti nei negozi sono associati a un valore `DATA_FINE_ASS` = *9999-12-31 00:00:00.0000000* e che ovviamente per negozi che vendono più moduli sono presenti più entry.

Anche in questo caso l'analisi svolta ha richiesto diverse operazioni prima di giungere a un dataset soddisfacente:

1. Poiché, come verrà descritto nella sezione 3.5, i dati delle vendite riguardano transazioni effettuate fino al 31 dicembre 2019, per mantenere uniformità nei dati sono stati ricercati (e rimossi) i moduli con una data di chiusura antecedente al 31/12/2019: fortunatamente quest'operazione non ha richiesto alcuna modifica al dataset;

Attributo	**Formato**	**Descrizione**
`unit_sk`	*intero*	Identificatore numerico del punto vendita, utilizzato come chiave nel database principale.
`POS_NEG`	*stringa (2)*	Sigla testuale che specifica in quale contesto urbano si trova il negozio (*CS*: centro storico, *CC*: centro commerciale, *PC*: periferia commerciale).
`TAB_MLI`	*stringa*	Identificativo del singolo modulo venduto, fornito nella forma testuale `MLIMxy`, dove `xy` è il numero definito nell'elenco puntato precedente.
`DATA_INIZIO_ASS`	*data*	Data di inizio vendita del modulo, nel formato ISO 8601.
`DATA_FINE_ASS`	*data*	Data di fine vendita del modulo, nel formato ISO 8601

Tabella 3.3: Gli attributi del dataset moduli.

unit_sk	POS_NEG	TAB_MLI	DATA_INIZIO_ASS	DATA_FINE_ASS
9036	CS	MLIM01	1988-07-01 00:00:00.0000000	9999-12-31 00:00:00.0000000
9036	CS	MLIM02	1988-07-01 00:00:00.0000000	9999-12-31 00:00:00.0000000
9036	CS	MLIM10	2014-01-01 00:00:00.0000000	9999-12-31 00:00:00.0000000
9036	CS	MLIM11	2014-01-01 00:00:00.0000000	9999-12-31 00:00:00.0000000
9075	PC	MLIM01	1994-10-01 00:00:00.0000000	9999-12-31 00:00:00.0000000
9075	PC	MLIM02	1994-10-01 00:00:00.0000000	9999-12-31 00:00:00.0000000
9075	PC	MLIM10	1994-10-01 00:00:00.0000000	9999-12-31 00:00:00.0000000
9075	PC	MLIM11	1994-10-01 00:00:00.0000000	9999-12-31 00:00:00.0000000

Tabella 3.4: Estratto delle prime righe dei moduli punti vendita.

2. Tramite **One-Hot Encoding** e successive elaborazioni sono state create per ogni entry del dataset cinque diversi attributi booleani (`M01`, `M02`, `M10`, `M11` e `M20`) per indicare la presenza o meno del modulo all'interno del punto vendita;
3. Sono state rimosse le colonne `DATA_INIZIO_ASS` e `DATA_FINE_ASS` in quanto non necessarie alle fasi successive di clustering e previsione vendite;
4. Sono state ricercate inconsistenze nei dati (valori `NaN` o negozi senza alcun modulo attivo) e rimosse eventuali entry errate: fortunatamente il dataset non è stato modificato da questa operazione.

3.3 Metri quadrati

Il terzo dataset contiene le metrature quadrate dei singoli negozi, e possiede una struttura molto semplice, disponibile in tabella 3.5.

Attributo	**Formato**	**Descrizione**
`unit_sk`	*intero*	Identificatore numerico del punto vendita, utilizzato come chiave nel database principale.
`MQ_TOT`	*numerico*	I metri quadrati occupati dal negozio.

Tabella 3.5: Gli attributi del dataset metri quadrati.

La fase di preprocessing è stata conclusa dopo aver controllato l'esistenza di valori nulli o negativi (non presenti all'interno del dataset). Le prime righe del dataset (su 199 totali) sono disponibili in tabella 3.6

unit_sk	MQ_TOT
9036	333.50
9075	628.82
9096	96.00
9182	142.50
9214	287.50

Tabella 3.6: Estratto delle prime righe del dataset metri quadrati.

3.4 Passaggi dei clienti

Il quarto dataset analizzato raccoglie alcune informazioni relative al flusso di clienti: le **entrate ed uscite registrate** dai tornelli all'entrata dei negozi e

i **passaggi rilevati** dalle telecamere di sorveglianza. I punti vendita Terranova sono dotati infatti di un sistema in grado di analizzare gli stream video per contare quante persone effettuano un passaggio davanti all'entrata. In questo modo possono essere eseguite previsioni e analisi sul numero di clienti *potenziali* e *reali*. La struttura del dataset è rappresentata in tabella 3.7, le cui prime righe sono disponibili nella tabella 3.8.

Attributo	**Formato**	**Descrizione**
`unit_sk`	*intero*	Identificatore numerico del punto vendita, utilizzato come chiave nel database principale.
`date_fk`	*data*	Data di riferimento dei valori.
`ingressi_rilev`	*intero*	Ingressi registrati dai tornelli.
`uscite_rilev`	*intero*	Uscite registrate dai tornelli.
`passaggi_rilev`	*intero*	Valore estratto dalle immagini delle videocamere di sorveglianza relativo al numero di potenziali clienti.

Tabella 3.7: Gli attributi del dataset passaggi.

unit_sk	date_fk	ingressi_rilev	uscite_rilev	passaggi_rilev
11212	2019-05-05	1768	1690	0
11212	2019-10-14	559	534	0
23351	2019-10-14	157	156	0
21019	2019-05-21	268	273	0
23351	2019-08-21	243	249	0

Tabella 3.8: Estratto delle prime righe dei passaggi nei punti vendita.

Questo dataset ha richiesto una profonda analisi delle distribuzioni dei valori e di pulizia delle entry inutili, in particolare:

1. Dopo un'attenta valutazione dell'attributo `passaggi_rilev` è stato rilevato un vuoto nei dati molto ampio, che ha portato alla rimozione dell'attributo a fine analisi poiché:
   (a) Su 62777 entry, 38240 presentavano un valore uguale a zero (il 60.91% del totale);
   (b) Raggruppando le righe per settimana, su 8370 previste 4889 erano sprovviste di valore (ovvero il 58.41%);

   (c) Rilassando il vincolo e raggruppando per mese, 1195 entry su 1937 previste erano composte da meno di 28 giorni (ovvero il 61.69%);

   (d) Infine, raggruppando solo per negozio (ottenendo quindi una media annuale), su 172 punti vendita 87 non presentavano alcun valore (il 50.58% del totale).

2. Le colonne `ingressi_rilev` e `uscite_rilev` sono state accorpate in un unico valore `movement`:

   (a) Tramite media aritmetica, se presenti entrambi;

   (b) Tenendo quello disponibile se uno dei due mancante;

   (c) Rimuovendo direttamente la riga se entrambi nulli;

3. Dall'attributo `date_fk` è stato estratto il mese e successivamente è stato eseguito un raggruppamento delle entry, ottenendo la media aritmetica mensile del valore `movement` per ogni punto vendita;

4. È stato analizzato nuovamente il dataset raggruppato per cercare eventuali discordanze nei dati, e sono emersi ben ventiquattro punti vendita (rappresentati in figura 3.1) con i seguenti problemi:

   (a) Alcuni negozi inaugurati durante l'anno presentavano diversi mesi contigui senza alcun valore;

   (b) Il negozio `25479` ha avuto un problema tecnico nel sistema di conteggio ai tornelli e ad Aprile ha riconosciuto una sola persona;

   (c) I negozi `15465` e `15523` hanno avuto problemi simili durante l'anno e per un mese intero non sono stati registrati passaggi.

5. Per risolvere il problema al punto precedente sono stati *riempiti* gli attributi nulli tramite **media pesata sull'andamento mensile** dei punti vendita: il risultato è disponibile in figura 3.2.

3.5 Storico delle vendite

Il quinto dataset è relativo alle vendite effettuate dai negozi in una particolare data. La composizione degli attributi è descritta in tabella 3.9, mentre è disponibile un estratto delle prime cinque righe nella tabella 3.10.

Attributo	**Formato**	**Descrizione**
`unit_sk`	*intero*	Identificatore numerico del punto vendita, utilizzato come chiave nel database principale.
`date_fk`	*data*	Data di riferimento del record.
`pz`	*intero*	Quantità di capi d'abbigliamento complessivi venduti nella specifica data.

Tabella 3.9: Gli attributi del dataset vendite.

unit_fk	date_fk	pz
21105	2019-09-15	146
24815	2019-07-17	460
23359	2019-02-05	238
19233	2019-11-07	516
16347	2019-11-05	117

Tabella 3.10: Estratto delle prime righe di storico vendite.

I dati non hanno richiesto particolari elaborazioni o modifiche (se non l'ordinamento per data), ma sono stati manipolati per poter creare **due ulteriori dataset**:

1. La **media delle vendite annuali** per ogni negozio (ai fini di clustering);
2. Le vendite medie **settimanali** per ciascun punto vendita (per poter percorrere diversi approcci durante la fase di forecasting).

Da una successiva analisi sono stati riscontrati record mancanti nel dataset, in grado di determinare negozi senza giorni (o settimane) di venduto. Questa situazione è stata causata da chiusure programmate dei punti vendita per svariati motivi (ampliamento negozio, chiusure forzate dei centri commerciali o combinazioni di feste nazionali e locali). Per questo motivo, dopo un confronto con il responsabile aziendale, i *vuoti* informativi sono stati mantenuti inalterati.

month	1	2	3	4	5	6	7	8	9	10	11	12
unit_sk												
15465	1834	1521	1545	1702	1695	1961	2253	1759	nan	2592	2351	2856
15523	2484	1582	1499	1945	1253	nan	2814	2403	1727	1558	1954	2375
16624	nan	nan	nan	nan	nan	nan	nan	nan	1028	1039	921	1286
23430	nan	nan	nan	nan	nan	nan	nan	nan	nan	755	306	557
23436	nan	nan	nan	nan	nan	nan	nan	nan	635	581	528	612
23438	nan	nan	nan	nan	888	736	835	784	665	645	746	931
23773	nan	nan	nan	nan	nan	nan	491	473	336	260	272	411
23796	nan	1286	612	504	429	475	615	537	542	484	503	697
24583	nan	nan	nan	nan	nan	nan	968	610	402	349	324	376
24584	nan	nan	nan	nan	2171	1917	1674	1227	1011	941	1031	1415
24813	nan	nan	nan	nan	nan	nan	nan	nan	nan	446	413	427
24815	nan	nan	326	264	216	203	250	268	220	188	184	235
24816	nan	nan	nan	456	238	205	248	233	222	182	176	230
24836	nan	nan	805	840	737	771	1015	916	600	627	674	821
25443	nan	nan	nan	nan	nan	nan	nan	nan	nan	425	364	418
25474	nan	nan	nan	nan	nan	208	579	444	309	301	289	330
25475	nan	nan	nan	nan	nan	1282	1693	1548	767	661	811	1056
25479	nan	nan	nan	1	318	290	306	305	247	235	246	281
25481	nan	nan	nan	nan	148	1199	1588	1799	1136	763	943	1447
25751	nan	nan	nan	nan	nan	nan	1350	903	692	574	549	728
25754	nan	nan	nan	nan	nan	nan	2914	2494	1613	1491	1861	2386
26481	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	1021	1318
26486	nan	nan	nan	nan	nan	nan	2247	1628	1364	996	1205	1490
26511	nan	nan	nan	nan	nan	nan	nan	nan	nan	443	453	537

Figura 3.1: Difetti nei valori dell'attributo `movement`.

unit_sk	tr_1	tr_2	tr_3	tr_4	tr_5	tr_6	tr_7	tr_8	tr_9	tr_10	tr_11	tr_12
15465	1834.0	1521.0	1545.0	1702.0	1695.0	1961.0	2253.0	1759.0	2006.0	2592.0	2351.0	2856.0
15523	2484.0	1582.0	1499.0	1945.0	1253.0	1963.0	2814.0	2403.0	1727.0	1558.0	1954.0	2375.0
16624	1186.0	898.0	885.0	1011.0	890.0	1047.0	1406.0	1227.0	1028.0	1039.0	921.0	1286.0
23430	604.0	457.0	451.0	515.0	453.0	533.0	715.0	624.0	503.0	755.0	306.0	557.0
23436	654.0	495.0	488.0	557.0	490.0	577.0	775.0	676.0	635.0	581.0	528.0	612.0
23438	928.0	703.0	693.0	792.0	888.0	736.0	835.0	784.0	665.0	645.0	746.0	931.0
23773	450.0	340.0	336.0	383.0	337.0	397.0	491.0	473.0	336.0	260.0	272.0	411.0
23796	608.0	1286.0	612.0	504.0	429.0	475.0	615.0	537.0	542.0	484.0	503.0	697.0
24583	607.0	460.0	453.0	518.0	455.0	536.0	968.0	610.0	402.0	349.0	324.0	376.0
24584	1696.0	1284.0	1266.0	1447.0	2171.0	1917.0	1674.0	1227.0	1011.0	941.0	1031.0	1415.0
24813	480.0	363.0	358.0	409.0	360.0	423.0	569.0	496.0	400.0	446.0	413.0	427.0
24815	268.0	203.0	326.0	264.0	216.0	203.0	250.0	268.0	220.0	188.0	184.0	235.0
24816	292.0	221.0	218.0	456.0	238.0	205.0	248.0	233.0	222.0	182.0	176.0	230.0
24836	889.0	673.0	805.0	840.0	737.0	771.0	1015.0	916.0	600.0	627.0	674.0	821.0
25443	450.0	341.0	336.0	384.0	338.0	397.0	534.0	466.0	375.0	425.0	364.0	418.0
25474	428.0	324.0	319.0	365.0	321.0	208.0	579.0	444.0	309.0	301.0	289.0	330.0
25475	1360.0	1029.0	1015.0	1160.0	1020.0	1282.0	1693.0	1548.0	767.0	661.0	811.0	1056.0
25479	332.0	251.0	248.0	283.0	318.0	290.0	306.0	305.0	247.0	235.0	246.0	281.0
25481	1344.0	1018.0	1003.0	1146.0	148.0	1199.0	1588.0	1799.0	1136.0	763.0	943.0	1447.0
25751	961.0	728.0	718.0	820.0	721.0	848.0	1350.0	903.0	692.0	574.0	549.0	728.0
25754	2557.0	1936.0	1909.0	2181.0	1919.0	2257.0	2914.0	2494.0	1613.0	1491.0	1861.0	2386.0
26481	1326.0	1004.0	990.0	1131.0	995.0	1171.0	1572.0	1372.0	1105.0	1027.0	1021.0	1318.0
26486	1790.0	1355.0	1336.0	1527.0	1343.0	1580.0	2247.0	1628.0	1364.0	996.0	1205.0	1490.0
26511	535.0	405.0	399.0	456.0	401.0	472.0	634.0	553.0	445.0	443.0	453.0	537.0

Figura 3.2: Correzione dei valori nulli per l'attributo `movement`.

3.6 I dati meteo

Il sesto e ultimo dataset analizzato è quello relativo ai dati meteorologici utilizzati da Microsoft Neural Network per svolgere le previsioni.

Dopo una analisi sommaria il dataset non è stato ritenuto valido poiché fornisce solo la temperatura media e massima, e non l'indicatore delle precipitazioni in millimetri. Sono stati quindi estratti i dati necessari da un servizio meteorologico locale (ovvero 3BMeteo), seguendo il seguente iter:

1. Partendo dall'anagrafica negozi della sezione 3.1 sono state convertite le coordinate geografiche in **località** (tramite il servizio di reverse-geocoding `Nominatim` di *OpenStreetMap*);
2. Sono stati corretti i nomi contenenti punteggiatura e spazi;
3. È stato utilizzato `BeautifulSoup` per estrarre dal portale del servizio i dati relativi al meteo locale dell'intero 2019;
4. Sono stati raggruppati i dati meteorologici locali per mese;
5. Infine, a ogni negozio è stato associato lo storico mensile del meteo della città di appartenenza.

Un estratto dei dati elaborati e raggruppati è disponibile nella tabella 3.11, per questioni di spazio non sono presenti tutte le colonne.

city	prec_1	prec_2	prec_3	...	t_mean_10	t_mean_11	t_mean_12
Agrigento	4.14	1.00	2.06	...	19.40	14.74	12.56
Alba	0.52	0.68	0.22	...	15.04	7.84	5.55
Alcamo	6.28	2.58	2.38	...	20.40	15.06	13.16
Alessandria	1.09	1.39	0.45	...	15.72	8.36	5.55
Amantea	9.38	3.28	2.26	...	20.63	16.87	13.60

Tabella 3.11: Estratto del dataset integrato da 3BMeteo.

In questo modo, alla fine dell'elaborazione ogni negozio è stato associato alle precipitazioni medie (`prec_1`, `prec_2`, etc) e alla temperatura media (`t_mean_1`, `t_mean_2`, etc), tramite l'attributo `city` elaborato da `Nominatim`.

3.7 Unione dei dati

È stato infine necessario, prima di poter eseguire clusterizzazione e previsione vendite, unire i vari dataset, rappresentati in figura 3.3 come schema *Entity Relationship*. Avendo dimensioni diverse sono stati rimossi gli elementi in

eccesso, ovvero le entry senza un riferimento `unit_sk` all'interno di **ogni dataset**. È stato per questo motivo mantenuto l'insieme più piccolo, ovvero quello relativo al flusso di persone (*Passaggi*) con 172 entry.

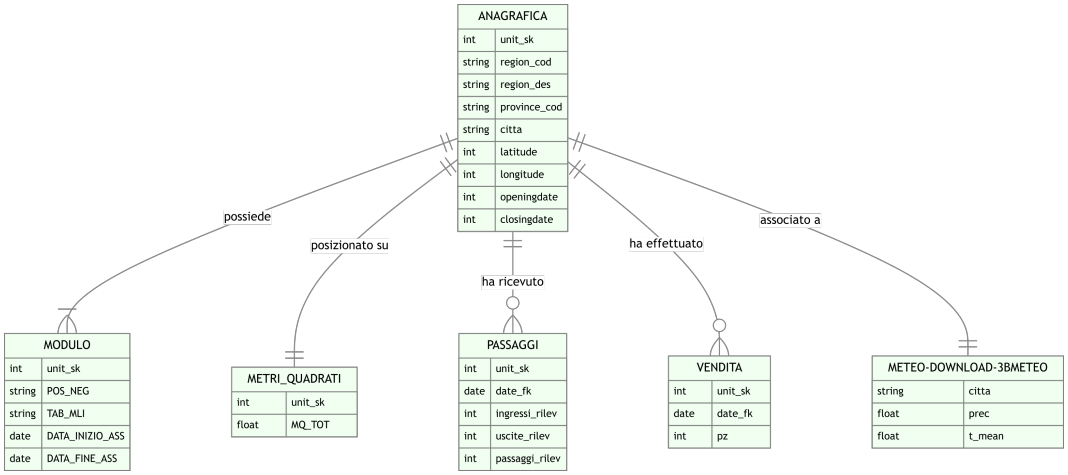


Figura 3.3: Diagramma entity relationship dei dataset analizzati.

Dopo aver scremato i risultati ed aver unito i punti vendita in un unico insieme, è stata eseguita nuovamente un'**analisi** volta a controllare eventuali discordanze o incongruenze. È emerso che:

- In tutti i 172 negozi rimasti i moduli *01* e *02* fossero sempre presenti, per tale motivo sono stati entrambi rimossi in quanto non discriminanti per la fase di clustering successiva;
- I moduli *10* e *11* fossero *speculari*: qualunque negozio con il primo modulo dispone anche del secondo e viceversa, per tale motivo è stato mantenuto solo *M10*;
- L'attributo `openingdate` è stato rimpiazzato con il relativo anno, per evitare un'esplosione dei possibili valori nel dataset;
- L'attributo relativo alla provincia `province_cod` fosse troppo dispersivo (93 valori) per poter essere utile in un'operazione clustering, ed è stato rimosso;
- L'attributo relativo alla regione `region_cod` fosse poco utile: per definizione di attributo categorico, un valore è diverso in egual modo da tutti gli altri, ma non esiste un metodo per esprimere la *vicinanza reale tra regioni* (come ad esempio Lombardia con Veneto rispetto a Sicilia e Molise), per questo motivo l'attributo è stato sostituito con un più generico `zone`, dove:

  - Al posto di Lombardia, Piemonte, Emilia Romagna, Valle D'Aosta, Veneto, Friuli-Venezia Giulia, Liguria e Trentino è stato utilizzato il valore **Nord**;
  - Al posto di Toscana, Lazio, Marche, Abruzzo e Umbria è stato usato il valore **Centro**;
  - Al posto di Puglia, Calabria, Campania, Molise e Basilicata è stato usato il valore **Sud**;
  - Al posto di Sicilia e Sardegna il valore **Isole**;

- Infine, poiché avere un attributo mensile per *precipitazioni*, *temperatura* e *clienti giornalieri* genera $12 \times 3 = 36$ attributi totali, il rischio è quello della **course of dimensionality** (esplosione della dimensonalità) ed è stato quindi elaborato un secondo dataset contenente **gli stessi attributi ma di periodo stagionale** (estate, primavera, autunno e inverno al posto dei mesi).

Tutti gli altri attributi analizzati non presentavano anomalie di alcun genere. Riassumendo quindi, entrambi i dataset finali (`monthly.csv` e `seasonal.csv`) contenevano:

Attributo	**Formato**	**Descrizione**
`unit_sk`	*intero*	Identificatore numerico del punto vendita, utilizzato come chiave nel database principale.
`zone`	*stringa*	Appartenenza geografica.
`latitude`	*intero*	Latitudine del punto vendita.
`longitude`	*intero*	Longitudine del punto vendita.
`openingdate`	*intero*	Anno di apertura.
`MQ_TOT`	*numerico*	Metratura quadrata del punto vendita.
`POS_NEG`	*stringa*	Locazione del negozio nel contesto urbano.
`M10`	*booleano*	Indicatore della presenza di capi d'abbigliamento per l'infanzia.
`M20`	*booleano*	Indicatore della presenza di una sezione per intimo all'interno del punto vendita.
`pz`	*numerico*	Venduto medio giornaliero del negozio.

Tabella 3.12: Gli attributi del dataset finale.

E in aggiunta un **set di attributi mensili o stagionali**:

- `tr_[1...12]` oppure `tr_[fall, winter, summer, spring]` per i visitatori giornalieri medi;
- `prec_[1...12]` oppure `prec_[fall, winter, summer, spring]` per la quantità di precipitazioni media;
- `t_mean_[1...12]` oppure `tr_mean_[fall, winter, summer, spring]` per la temperatura media.

Capitolo 4

I risultati raggiunti

In questa sezione finale della tesi vengono descritti i risultati conseguiti nelle fasi di clusterizzazione e forecasting. Si analizzeranno gli errori generati dai diversi algoritmi, confrontando in secondo luogo le prestazioni del sistema attualmente utilizzato da Teddy con quelli studiati all'interno di questa tesi.

4.1 Clustering

Il primo aspetto che verrà affrontato è quello relativo al clustering dei punti vendita, poiché i risultati di questa operazione sono stati poi fondamentali per le scelte intraprese nella fase di previsione vendite.

4.1.1 Dashboard Streamlit

Per velocizzare l'analisi clustering è stata sviluppata una dashboard interattiva basata sul framework web Streamlit[1] in grado di:

- Automatizzare la creazione e l'esecuzione degli algoritmi di clustering (K-means, gerarchico, etc);
- Velocizzare la scelta degli attributi da mantenere nel dataset;
- Visualizzare graficamente l'andamento dei principali indicatori di bontà dei cluster generati;
- Permettere l'analisi delle distribuzioni dei cluster.

[1]Framework di prototipazione rapido basato su Python: `https://streamlit.io/`

Benché abbia richiesto tempo per poter essere sviluppato correttamente, questo strumento ha permesso di focalizzarsi sui **risultati** del clustering, piuttosto che sui modi per ottenerli o sul codice necessario per sviluppare i modelli. L'interfaccia, visibile in figura 4.1, è stata suddivisa in due parti:

- Una barra laterale completamente dedicata alle **impostazioni** del dataset e ai parametri relativi all'algoritmo utilizzato;
- La pagina principale contenente tutti gli indicatori d'errore, le misure di bontà dei cluster e diversi diagrammi e grafici utili a capire i **risultati della clusterizzazione** svolta.

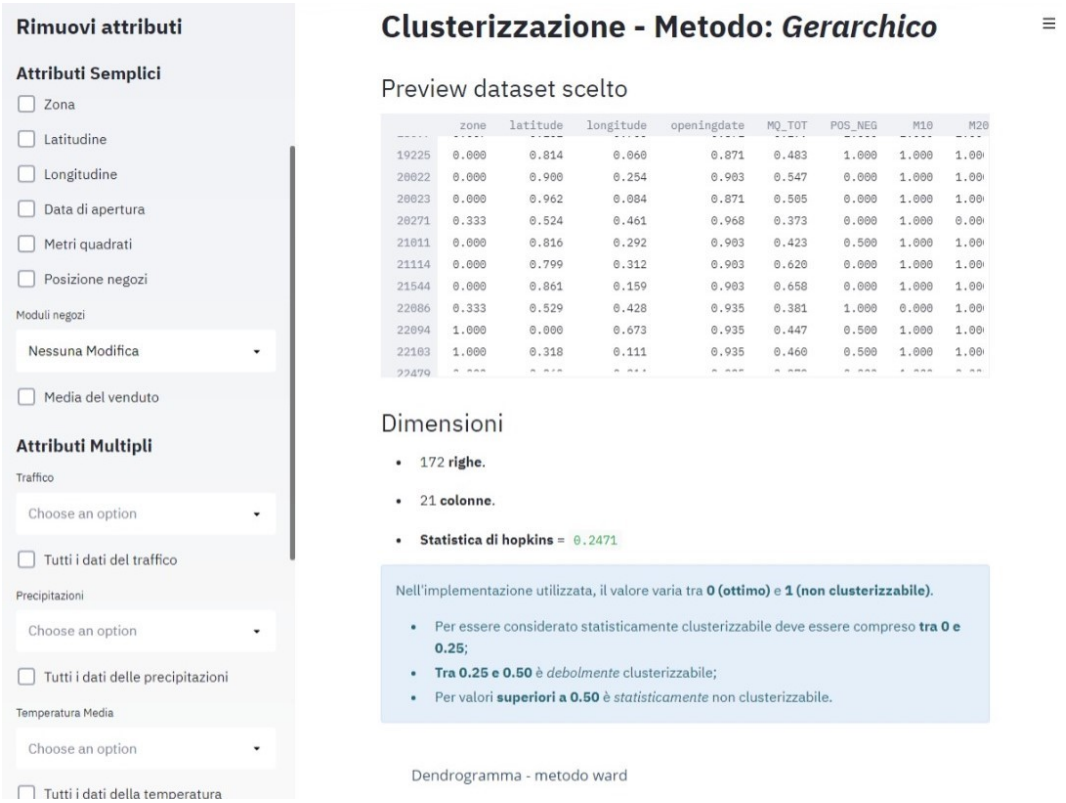
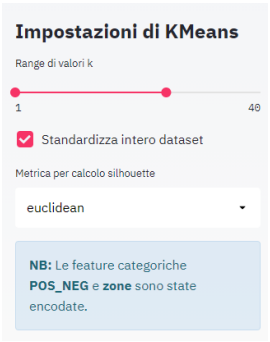


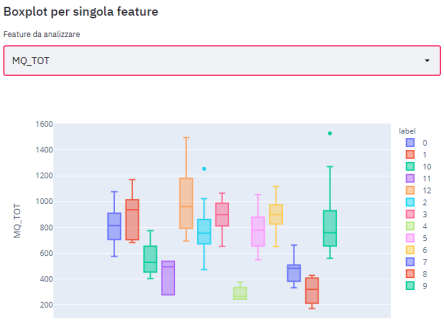
Figura 4.1: Schermata iniziale della dashboard Streamlit sviluppata.

In questo modo è stato possibile seguire il processo di analisi concentrandosi prima sul dataset (stagionale o annuale) e sui metodi di manipolazione degli attributi, per poi scegliere l'algoritmo di clustering e visualizzarne graficamente i risultati. All'interno dell'immagine 4.2 sono rappresentati alcuni dei componenti grafici che la dashboard fornisce.

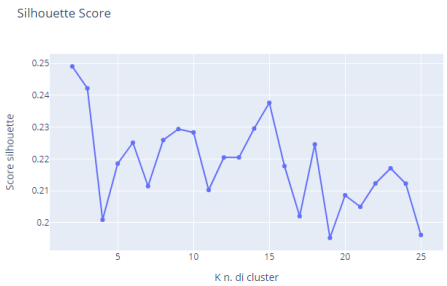


(a) Box impostazioni algoritmo clustering.

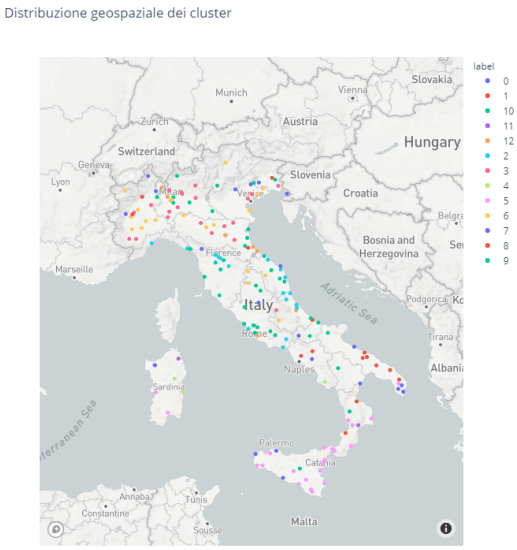
(b) Analisi delle distribuzioni dei cluster in base a due attributi da confrontare.



(c) Boxplot dei cluster in base a un attributo.



(d) Andamento silhouette in base al valore assunto da k.



(e) Distribuzione geospaziale dei negozi.

Figura 4.2: Alcuni elementi grafici della dashboard Streamlit.

L'interfaccia si **adatta** (ad esempio richiedendo l'inserimento dei parametri corretti per l'algoritmo scelto come in figura 4.2a) e permette di controllare diversi aspetti della clusterizzazione svolta: sia l'**errore** statistico generato (4.2d) che le varie **distribuzioni** dei punti (dal punto di vista geospaziale in 4.2e e di intervallo in 4.2c). Fornisce infine la possibilità di analizzare le **correlazioni** tra due attributi e i cluster formati (in figura 4.2b) per poter estrarre al meglio il significato dei singoli gruppi di elementi.

Il principale vantaggio di questo approccio è la possibilità di testare velocemente le variazioni dei risultati sulla base dei cambiamenti al dataset di partenza, passando dai risultati di una configurazione alla modifica immediata dei dati. Anche delegare la generazione delle misure di errore ha permesso di concentrarsi **unicamente sulla migliore riuscita possibile** della clusterizzazione.

4.1.2 Le analisi svolte

I primi tentativi sono stati svolti tramite K-means, in quanto è il più semplice di tutti gli algoritmi presi in esame e può essere usato per capire a grandi linee che tipo di risultati ci si può aspettare. Sin dalla prima esecuzione i dati sono stati **standardizzati** (i valori di ciascun attributo sono stati traslati su un intervallo tra 0 e 1) per poi essere sottoposti all'algoritmo di clustering, così da evitare che attributi relativi a valori più grandi influenzassero negativamente la riuscita della clusterizzazione. Dopo un'attenta analisi è stato possibile affermare che:

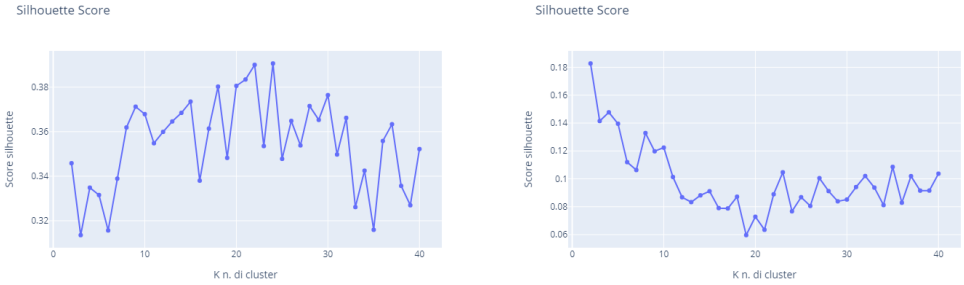
1. Il dataset con i dati **mensili** (numero clienti, media temperature e media precipitazioni) è risultato **altamente non clusterizzabile** a causa dei troppi attributi e relativa difficoltà ad identificare il numero di cluster;

2. Riducendo la dimensionalità tramite l'utilizzo del dataset stagionale, il valore di Hopkins è risultato identico a quello del dataset mensile, a causa del comunque cospicuo numero di attributi;

3. Mantenendo **un solo attributo** per ogni feature stagionale (temperatura, precipitazioni e clienti) e riducendo a 12 il numero di attributi totali, il dataset è risultato leggermente più clusterizzabile, ma comunque con una indice di Hopkins molto basso per poter essere considerato, stesso discorso rimuovendo l'anno di apertura e le coordinate: lo score silhouette massimo raggiunto da questa configurazione si è attestata sullo 0.30 se misurato tramite distanza euclidea o di Manhattan, 0.12 tramite distanza di Mahalanobis;

Sono stati poi rimossi i rimanenti attributi stagionali relativi al numero di clienti e al venduto medio, ma anche in questo caso sia il valore di silhouette

che la statistica di Hopkins non hanno subito una variazione significativa, segno del fatto che anche con questa configurazione il dataset non fosse clusterizzabile. Non sono stati rimossi ulteriori attributi per evitare di spogliare completamente il dataset da ogni sua caratteristica. I relativi andamenti del valore di silhouette sono disponibili in figura 4.3, dai quali risulta chiaramente che **non esiste** un valore k in grado di rappresentare in maniera adeguata i cluster. Da un grafico di questo tipo ci si aspetta generalmente un valore k *relativamente piccolo* che raggiunge un buon valore di silhouette (attorno a 0.80 circa per la distanza euclidea), che equivale a cluster ben separati e coesi. Se ciò non avviene, il dataset **potrebbe non essere adatto a essere clusterizzato**.



(a) Distanza euclidea. (b) Distanza di Mahalanobis.

Figura 4.3: Andamento del valore di silhouette in base al numero di cluster k.

In secondo luogo è stato impiegato un **approccio gerarchico**, utilizzando il metodo di Ward per calcolare la distanza intracluster. Sono state eseguite le stesse prove svolte con k-means fino a raggiungere l'ultima configurazione: il dendrogramma generato, visibile in figura 4.4, ha suggerito $k = 3$ come il numero di cluster.

Sfortunatamente una successiva analisi delle distribuzioni dei valori ha fatto emergere la completa eterogeneità dei cluster, per niente separati e distinti. Un esempio di questa situazione è riportata in figura 4.5, dove sono stati confrontati i metri quadrati dei negozi con il venduto medio.

Non ottenendo grandi risultati con la distanza euclidea, è stata impiegata la matrice delle **distanze di Gower** tramite algoritmo gerarchico. È stato utilizzato questo approccio in quanto, spesso i dati con varie feature categoriche ottengono risultati migliori in termini di silhouette rispetto agli approcci di encoding classici, come descritto esaustivamente in [SZ15]. Mantenendo il metodo di Ward come misura di distanza intracluster, è stato raggiunto un valore di silhouette molto più alto rispetto ai tentativi precedenti, il cui grafico è disponibile in figura 4.6.

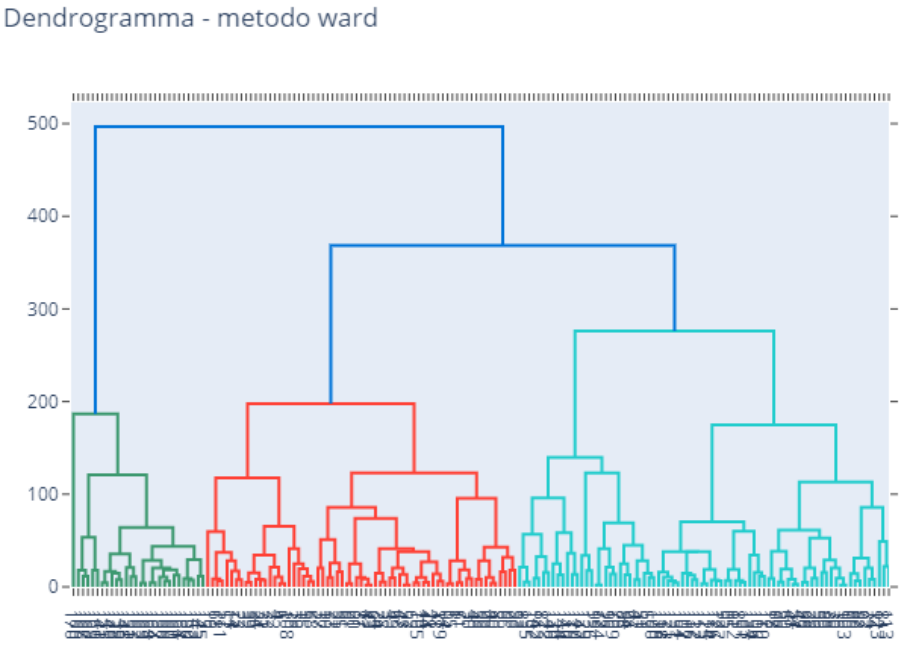


Figura 4.4: Dendrogramma generato dall'approccio gerarchico.

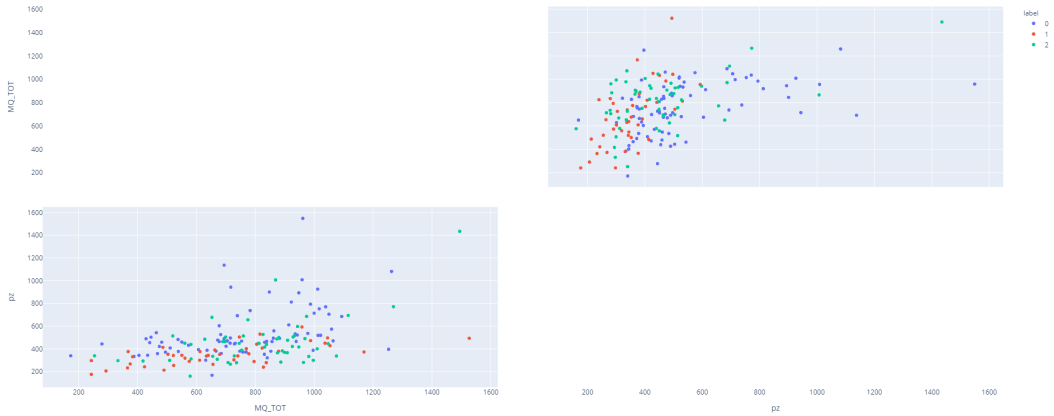


Figura 4.5: Un esempio di distribuzione dei punti secondo $k = 3$.

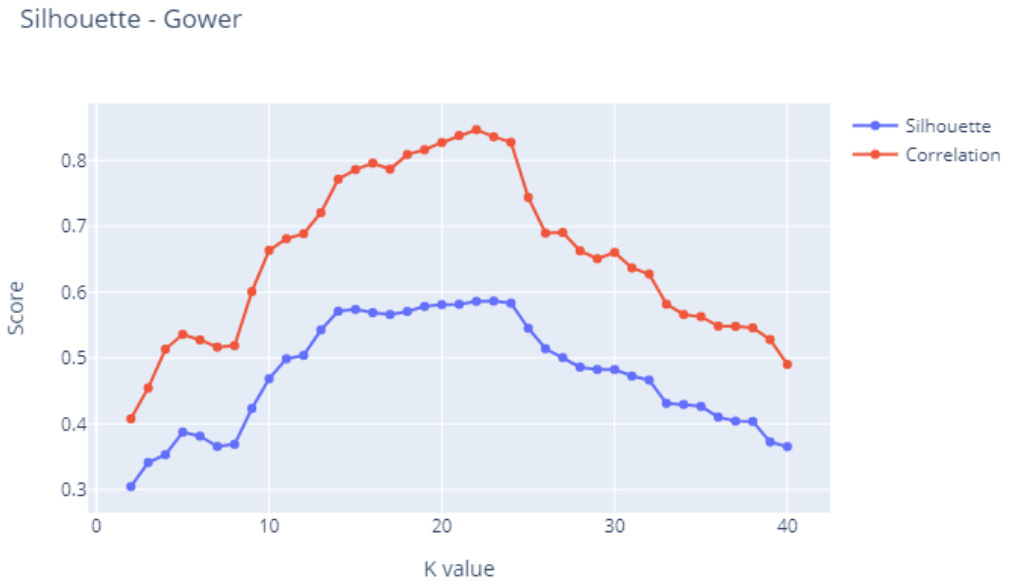


Figura 4.6: Andamento del valore di silhouette, tramite distanza di Gower.

Sfortunatamente, però, benché il valore di silhouette sia risultato decisamente migliore rispetto a quelli ottenuti con i precedenti metodi:

- Il numero di cluster indicato dal grafico (14) è risultato troppo grande per il tipo di clusterizzazione svolta e per la dimensione del dataset;
- Un valore di silhouette vicino a 0.50 non è comunque abbastanza per poter asserire di aver clusterizzato correttamente dei dati;
- La differenza tra correlazione (in rosso nel grafico) e silhouette (blu) è indice di cluster ben coesi ma poco separati un dall'altro, confermando il sospetto generato dai metodi precedenti (k-means e approccio gerarchico).

L'ultimo tentativo è stato fatto utilizzando l'implementazione dell'algoritmo DBSCAN, che è sfortunatamente risultato **troppo sensibile** per ogni possibile valore di *minPts* o ϵ. Essendo tutti i punti molto vicini tra loro, per qualunque valore di ϵ l'algoritmo li ha sempre raggruppati in un unico cluster, a eccezione di qualche noise-point. Con un valore ϵ tendente allo zero, DBSCAN ha categorizzato ogni elemento come noise-point.

4.1.3 I risultati

Visti i risultati ottenuti nelle fasi descritte precedentemente, il dataset è stato classificato come **difficilmente clusterizzabile**: l'unica configurazione utile ad ottenere un valore di Hopkins quantomeno valido (0.8789) include soltanto i dati relativi a zona geografica, contesto urbano del negozio, metri quadrati e moduli posseduti. Benché questo indicatore risulti favorevole, è necessario considerare anche che non esiste un valore k per cui la silhouette subisce un miglioramento netto rispetto ai successivi valori, come rappresentato in figura 4.7, ma al contrario l'andamento incrementale di k lo migliora in modo costante.

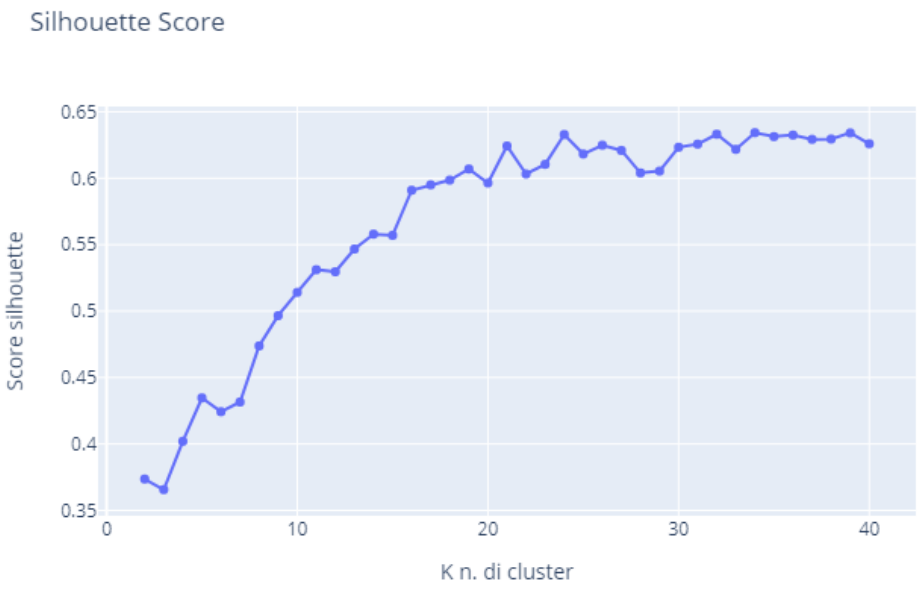


Figura 4.7: Il valore di silhouette migliore ottenibile.

Questo tipo di risultato è comune quando i dati iniziali si presentano molto *vicini* gli uni con gli altri, e risulta complesso raggrupparli in insiemi separati a causa della vicinanza.

Anche dopo un confronto con i responsabili aziendali, non sono emersi indicatori aziendali (KPI, *Key Performance Indicator*) in grado di attribuire un significato reale ai cluster generati.

4.2 Previsione delle vendite

Poiché la clusterizzazione non ha fornito alcun tipo di indicazione preliminare sui punti vendita, non è stato possibile utilizzare algoritmi in grado di supportare la cosiddetta eventualità di **cold start** (letteralmente *partenza a freddo*). Per poter sviluppare previsioni storiche su serie ridotte o addirittura vuote sono stati messi a punto negli anni numerosi algoritmi (come ad esempio Amazon Forecast) che però necessitano di una tabella di riferimento di similarità su cui basarsi. *Per prevedere le vendite di un articolo che sta per uscire sul mercato, ci si può basare sulle vendite passate di un articolo simile, ma è necessario sapere a cosa riferirsi.* Per questo motivo sono stati considerati solo due approcci **auto_arima** e **Prophet**.

Entrambi non supportano alcun tipo di meccanismo di *cold start*, per cui ci si è limitati a eseguire previsioni anche per negozi con pochissimi dati, cercando di minimizzarne comunque l'errore. Su quest'ultimo tipo di serie Microsoft Neural Network ottiene dei risultati **decisamente scadenti**, poiché sono necessari almeno due anni di storico per allineare previsioni e realtà.

4.2.1 Auto_arima

Visto che ogni serie storica potrebbe avere alcune componenti proprie (trend, stagionalità o altro ancora) è stato impiegato auto_arima per la ricerca della configurazione ottima di iperparametri **per ogni punto vendita**. Dall'analisi effettuata sui dati a disposizione è emerso che:

- Questo approccio, utilizzato su **dati settimanali**, ha ottenuto buoni risultati su larga parte dei negozi con uno storico consistente (**un anno intero di dati**);
- È molto complesso far funzionare l'algoritmo senza almeno un intero ciclo di dati: con una stagionalità annuale non possono essere eseguite previsioni accurate con pochi mesi di storico, ed è quindi necessario cambiare la stagionalità, portando però l'**errore anche al 30%** del valore reale, ritenuto dal punto di vista aziendale troppo alto;
- In linea di massima, **con un unico anno di training** sul 94% dei negozi l'RMSE annuale si è assestato su valori totalmente accettabili dal punto di vista aziendale (ovvero circa dal 3% al 15% rispetto a quanto preventivato dal modello);
- Per una piccola percentuale di negozi (circa il 7% del totale) auto_arima non è riuscito ad estrarre una configurazione di iperparametri in maniera automatica, ed è stato necessario selezionarla manualmente.

Un estratto dei risultati ottenuti da auto_arima su tre negozi è disponibile in tabella 4.1: anche l'errore ottenuto sul negozio `17368`, più alto di molti altri, è stato ritenuto dal responsabile aziendale accettabile anche nel caso peggiore (RMSE=251) poiché sono errori quantitativi assolutamente assimilabili dal personale dei punti vendita durante un'intera settimana.

Id	**Anni di training**	**RMSE totale**	**RMSE primo semestre**	**RMSE secondo semestre**
17368	5	132	109	155
	4	139	164	110
	3	139	159	117
	2	208	217	199
	1	251	289	213
18054	1	253	255	251
	2	113	85	130
	3	107	91	118
	4	129	118	137
	5	105	61	132
16252	6	43	34	50
	5	44	34	51
	4	70	61	77
	3	54	44	61
	2	48	37	55
	1	80	82	78

Tabella 4.1: Un estratto dei risultati ottenuti da auto_arima.

Il reale problema di questo approccio è la **complessità computazionale** nella ricerca della migliore configurazione di parametri, oltre al **tempo** richiesto dall'implementazione di auto_arima considerato troppo elevato dall'azienda.

4.2.2 Prophet

Le prove svolte tramite l'algoritmo Prophet hanno invece fatto emergere numerosi pregi rispetto ai risultati ottenuti in precedenza:

- Il modello può funzionare anche con serie **storiche parziali** (ad esempio su punti vendita appena aperti con qualche settimana di storico dove auto_arima non riesce ad essere usato) anche se **l'errore è decisamente elevato** in questi casi;

- Può essere addestrato in modo agevole su **stagionalità non complete** (ad esempio un anno e mezzo di dati) e fornire comunque previsioni accurate;
- È **molto veloce**, richiede qualche secondo per elaborare il modello di una singola serie storica;
- Con **un solo anno di training** è possibile ottenere una previsione più accurata del 3%-15% rispetto ad auto_arima, ma con una frazione del tempo richiesto;
- Con cinque o sei anni di training, sia auto_arima che Prophet ottengono praticamente **gli stessi risultati**, ma quest'ultimo rimane comunque molto più veloce dal punto di vista computazionale (un estratto comparativo tra i due algoritmi è disponibile nella tabella 4.2).

Id	Anni di training	RMSE auto_arima	RMSE Prophet
17368	5	132	105
18054	6	105	84
16252	6	43	42

Tabella 4.2: Comparativa errori tra Prophet e auto_arima.

4.2.3 Confronto con Microsoft Neural Network

Dopo aver confrontato i due algoritmi tra loro, è stato analizzato dal gruppo di data science interno all'azienda la performance di Prophet rispetto ai risultati raggiunti da Microsoft Neural Network. Prendendo in considerazione previsioni relative al solo anno 2019:

- Prophet nel 55.57% dei negozi **ha prodotto stime più affidabili**, con una deviazione standard più bassa (1987.08 Prophet, 5313.29 MNN) che si traduce in una distribuzione più centrata attorno alla media e quindi una propensione minore a commettere errori *elevati*;
- Nel 80.5% dei negozi considerati il test di bontà di adattamento (ovvero la *distanza* teorica tra le quantità stimate da un modello e i dati reali) di Prophet ha dato risultati migliori;
- Nel 61.1% dei negozi l'errore di previsione (RMSE) è risultato minore rispetto al modello Microsoft.

In figura 4.8 è disponibile la comparazione degli errori tra MNN e Prophet, mentre in figura 4.9 l'andamento delle previsioni tra i due algoritmi su due serie storiche diverse.

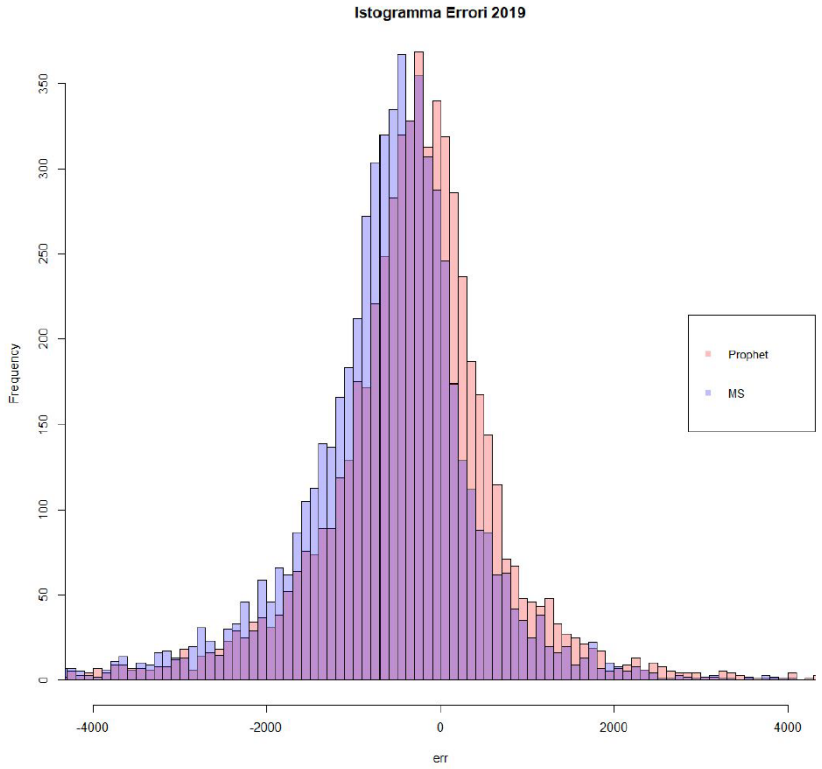


Figura 4.8: Istogramma di confronto degli errori tra MNN e Prophet.

Utilizzando invece lo stesso modello senza particolari accorgimenti **sull'anno 2020**, MNN ottiene dei risultati migliori, per la precisione nel 62.2% dei casi. Questo è derivato dal fatto che una rete neurale è molto più robusta a eventi esterni completamente slegati dai vari trend che un modello statistico come Prophet può utilizzare per le proprie previsioni. È però possibile calibrare i parametri del modello in modo tale da migliorare il valore di errore (RMSE) e cercare di utilizzare variabili esogene in grado di migliorare la previsione anche in condizioni complesse come nel caso attuale di un'emergenza sanitaria. Nei prossimi mesi, il team data science aziendale si concentrerà sul miglioramento del modello Prophet, in modo da procedere all'affiancamento graduale al sistema Microsoft Neural Network.

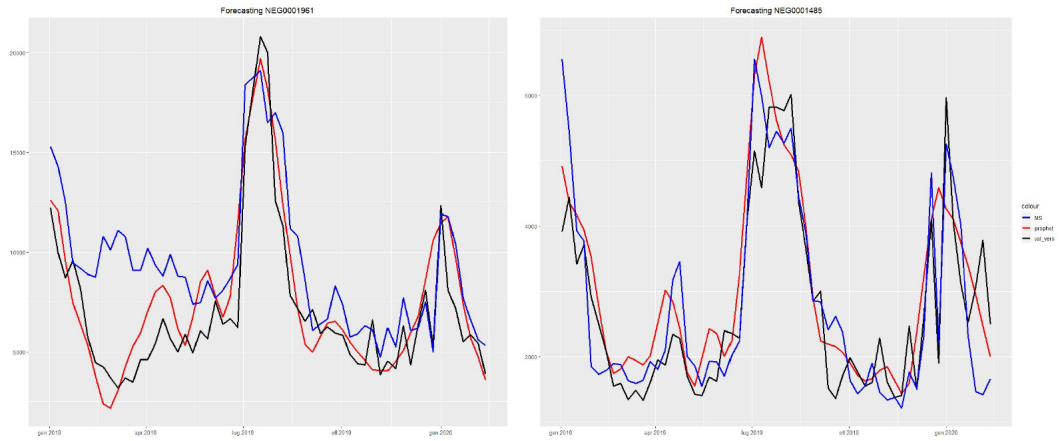


Figura 4.9: Differenza previsionale tra MNN (blu), Prophet (rosso) e il valore reale (nero).

Conclusioni

All'interno di questa tesi sono stati analizzati e gestiti alcuni dei dati relativi al marchio di abbigliamento Terranova tramite le principali tecniche di data mining, con lo scopo di estrarre informazioni utili da essi. I dati forniti sono stati analizzati in maniera approfondita e di conseguenza preprocessati: sono stati rimossi gli attributi non necessari e elaborati quelli utili secondo le principali tecniche di gestione del dato.

Per la fase di clustering sono stati generati più dataset di output, così da poter seguire diversi approcci tramite vari algoritmi di clustering: k-means, DBSCAN, agglomerativo gerarchico e tramite calcolo della matrice delle distanze di Gower. I risultati conseguiti da quest'ultimo approccio hanno raggiunto il più alto valore di silhouette, che è stato ritenuto comunque troppo basso: i dati forniti dal gruppo non sono stati quindi sufficienti a sviluppare un corretto modello di clustering.

Successivamente sono state analizzate le vendite dei negozi del brand Terranova, con lo scopo di eseguire una corretta previsione delle vendite tramite algoritmi statistici. Poiché la clusterizzazione ha dato esito negativo non è stato possibile utilizzare l'output di questa fase per gestire i punti vendita con pochi dati, su cui è complesso eseguire corrette previsioni dell'andamento. I principali approcci utilizzati all'interno di questa fase hanno coinvolto la famiglia dei modelli SARIMAX e l'algoritmo open source Prophet sviluppato da Facebook, entrambi basati su modelli additivi. Il principale vantaggio di questi approcci rispetto a più complesse e precise reti neurali è la possibilità di poter comprendere il perché delle previsioni generate, che si basano su diversi trend (annuali, settimanali o mensili) e possono essere analizzate nella loro interezza. I risultati migliori sono stati raggiunti da Prophet, anche se in linea di massima all'aumentare dello storico di vendita i due modelli convergono verso lo stesso errore. Il principale vantaggio dell'approccio sviluppato dal team Core Data Science di Facebook è il tempo richiesto per l'addestramento, esiguo rispetto a quello necessario ad auto_arima per la ricerca dei parametri iniziali ottimali e successivo forecast. Sulla base di indagini preliminari svolte, Prophet è risultato estremamente valido anche nei confronti dell'attuale algoritmo di previsione che il Gruppo Teddy utilizza: Microsoft Neural Network. Prophet

raggiunge risultati migliori sulla previsione dell'andamento delle vendite per l'anno 2019, ma la robustezza di una rete neurale come Microsoft Neural Network è determinante quando si eseguono previsioni sul 2020. La situazione sanitaria legata alla pandemia globale viene gestita meglio rispetto a Prophet.

Per migliorare i risultati conseguiti in entrambe le fasi di clustering e previsione possono essere presi diversi provvedimenti applicabili sia ai modelli che ai dati forniti. In primo luogo si potrebbe provare ad eseguire nuovamente la clusterizzazione con altri dati o indicatori economici dei negozi, visto che le informazioni canoniche sui punti vendita non hanno identificato dei cluster coesi e ben separati. Per quanto riguarda la fase di previsione delle vendite possono essere presi provvedimenti su due fronti diversi: migliorare il forecasting per negozi con pochi dati (cold start) e ottimizzare le previsioni per l'anno 2020 adattando il modello alla situazione pandemica in corso. Nel primo caso si potrebbero clusterizzare i negozi utilizzando soltanto le informazioni sul venduto della prima settimana o del primo mese, ottenendo dei cluster basati su quell'indicatore economico. Poiché Prophet supporta le variabili esogene, se si trovassero dei gruppi omogenei e separati potrebbero essere utilizzate le serie storiche dei punti vendita simili come variabili esterne. Queste indicazioni dovrebbero portare al miglioramento delle previsioni per negozi appena aperti, forse con errori comunque alti, ma sicuramente più utilizzabili di quelli raggiunti tramite approccio classico.

Per il miglioramento delle previsioni sull'anno 2020 è necessario studiare quali indicatori permettono di migliorarne i risultati e quali variabili esogene associare (ad esempio il peggioramento o miglioramento del quadro sanitario e pandemico). È difficile che un modello statistico riesca a raggiungere le prestazioni di una rete neurale in queste situazioni, ma è auspicabile che la previsione possa raggiungere dei livelli quanto meno accettabili dal punto di vista dell'errore. Essendo una previsione non esattamente critica dal punto di vista aziendale (sono accettati errori anche di 100 o 200 unità settimanali) è possibile che anche un modello come Prophet riesca a fornire le giuste previsioni se ottimizzato a dovere.

Bibliografia

[Mah36] Prasanta Chandra Mahalanobis. «On the generalized distance in statistics». In: National Institute of Science of India. 1936.

[Roy19] Baijayanta Roy. *All about Categorical Variable Encoding*. Lug. 2019. URL: `https://towardsdatascience.com/all-about-categorical-variable-encoding-305f3361fd02`.

[Gow71] John C Gower. «A general coefficient of similarity and some of its properties». In: *Biometrics* (1971), pp. 857–871.

[Est+96] Martin Ester et al. «A density-based algorithm for discovering clusters in large spatial databases with noise.» In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[HS54] Brian Hopkins e John Gordon Skellam. «A new method for determining the type of distribution of plant individuals». In: *Annals of Botany* 18.2 (1954), pp. 213–227.

[HA18] Rob J Hyndman e George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[Box+15] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[TL18] Sean J Taylor e Benjamin Letham. «Forecasting at scale». In: *The American Statistician* 72.1 (2018), pp. 37–45.

[She00] Colin Shearer. «The CRISP-DM Model: The New Blueprint for Data Mining». In: *Journal of Data Warehousing* 5.4 (2000).

[SZ15] Tiago RL dos Santos e Luis E Zárate. «Categorical data clustering: What similarity measure to recommend?» In: *Expert Systems with Applications* 42.3 (2015), pp. 1247–1260.