Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Matematica

ARMONIZZAZIONE DEI DATI PER L'ADDESTRAMENTO DI RETI NEURALI RICORRENTI: APPLICAZIONE PER LA GESTIONE DELLE PROMOZIONI NEL SETTORE RETAIL

Tesi di Laurea in Ricerca Operativa

Relatore:
Chiar.mo Prof.
Daniele Vigo

Presentata da:
Maurizio Cesaroni

Correlatori:
Dott. Carlo Del Castello
Dott. Tiziano Parriani

III Sessione
Anno Accademico

Introduzione

Lo sviluppo di questa tesi è stato reso possibile dalla collaborazione con l'azienda Kantar - Consulting presso la quale ho svolto il tirocinio curricolare. Quest'ultimo ha contribuito a definire il problema e soprattutto a validare i risultati nella sperimentazione.

Scopo di questa tesi è stato lo sviluppo di un modello matematico per la convalida, modifica o eliminazione automatica dei dati in possesso dell'azienda al fine di migliorare il set che sarà poi alla base della rete neurale predittiva in uso. Problemi di questo tipo vengono detti di Data Harmonization, rappresentano un argomento di grande interesse per le aziende in quanto spesso i database contengono dati inconsistenti, sbagliati o incompleti che possono portare a cattive interpretazioni o decisioni.

La tesi è strutturata come segue:

- Il Capitolo 1 introduce la ricerca operativa e fornisce i principali risultati nel caso specifico della programmazione lineare mista-intera di cui il modello sviluppato farà parte. Inoltre, daremo una panoramica del software utilizzato per la risoluzione del modello.

- Nel Capitolo 2 tratteremo solo concettualmente il tema delle reti neurali per arrivare a trattare il caso specifico delle Long short-term memory (LSTM) che rappresenteranno il metro di giudizio del nostro modello.

- Nel Capitolo 3 daremo spiegazione della Data Harmonization, individuandone lo scopo, facendo riferimento alle tecniche più recenti e ad alcuni dei differenti approcci utilizzati.

- Nel Capitolo 4 andremo a definire quello che è l'ambiente Kantar-Consulting, presso il quale ho effettuato il tirocinio, andando a dare spiegazione di alcuni elementi interni al processo produttivo dell'azienda che ci interessano per capire le modellazioni effettuate.

- Nel capitolo 5, daremo spiegazione del problema affrontato e degli obiettivi del lavoro proposto. Dopo una rapida rassegna dei modelli preliminari sviluppati, enunceremo il modello finale andandone a spiegare le varie componenti.

- Nel capitolo 6 presenteremo i risultati del modello finale in relazione al LSTM, analizzando alcuni casi specifici e osservando il tipo di scelte attuate dal modello.

- Nel capitolo 7 daremo alcuni spunti per eventuali migliorie da apportare al modello o processi che potrebbero migliorarne i risultati.

Indice

Capitolo 1

Elementi di ricerca operativa

In questo capitolo introduciamo la ricerca operativa e i principali risultati nell'ambito dell'ottimizzazione lineare che useremo in seguito per la formulazione del modello e la sua comprensione.

1.1 Ricerca Operativa

La ricerca operativa (Operations Research OR) è la branca della matematica applicata in cui problemi decisionali complessi vengono analizzati e risolti mediante modelli matematici e metodi quantitativi avanzati. Permette di operare la scelta della configurazione migliore per raggiungere un determinato obiettivo rispettando vincoli che sono imposti dall'esterno e non sono sotto il controllo di chi deve compiere le decisioni.

Si occupa dunque di formalizzare un problema in un modello matematico e calcolare una soluzione ottima, quando possibile, o approssimata detta anche subottima. Ha molte applicazioni commerciali soprattutto negli ambiti economico, infrastrutturale, logistico, militare, della progettazione di servizi e di sistemi di trasporto e nelle tecnologie.

I modelli matematici di un problema di ottimizzazione o decisione, sono caratterizzati da tre componenti:

- un set di **variabili decisionali** che rappresentano le configurazioni possibili da prendere. Sono espresse mediante un vettore di dimensione n, $x \in \mathbb{R}^n$, e possono essere continue, discrete o binarie.

- una **regione ammissibile** $F \subseteq \mathbb{R}^n$ che include le configurazioni di variabili decisionali che sono ammissibili per il problema. La regione può essere definita esplicitamente, elencando tutte le possibili configurazioni, o implicitamente, attraverso l'utilizzo di vincoli determinati da equazioni e/o disequazioni che le configurazioni ammissibili devono rispettare. Se $F = \emptyset$ il problema risulta impossibile.

- una **funzione obiettivo**, $\varphi : F \to \mathbb{R}$ che associa un valore ad ogni configurazione ammissibile. In base al caso di studio si è interessati a massimizzare o minimizzare questo valore.

Abbiamo quindi la seguente forma:

$$min\ (max) \quad \varphi(x)$$

$$\begin{aligned} s.t. \quad & g_i(x) \leq 0 \quad \forall i = (1, ..., m) \\ & h_j(x) = 0 \quad \forall j = (1, ..., p) \end{aligned}$$

$$x \in \mathbb{R}^n$$

I modelli di programmazione possono essere classificati in base alla forma delle loro variabili, dei loro vincoli e della funzione obiettivo:

- φ, g_i, h_j qualunque, si parla di Programmazione Non Lineare (NLP) per cui non si hanno algoritmi generali, esistono metodi con cui è possibile convergere ad ottimi locali.

- φ, g_i convesse e h_j lineari, abbiamo un problema di Programmazione Convessa (CP) in cui l'ottimo locale coincide con quello globale, esistono algoritmi per la risoluzione ma non sono efficienti in termini di tempo.

- φ, g_i, h_j lineari, allora siamo davanti ad un problema di Programmazione Lineare (LP) in cui ottimo locale e globale coincidono e abbiamo algoritmi efficienti per la risoluzione come l'algoritmo del simplesso e dell'ellissoide.

- Se siamo nel caso di un problema di Programmazione Lineare e inoltre tutte le variabili sono intere, si parla di Programmazione Lineare Intera (ILP). Se vi sono variabili continue e intere, avremo un problema di Programmazione Lineare Mista-Intera (MILP).

Quindi, in generale la soluzione di un problema di ottimizzazione in cui g_i, h_j sono funzioni qualunque non risulta facile. In quanto non si dispone di algoritmi in grado di trovare una soluzione in un tempo ragionevole, questo porta a sviluppare modelli semplificati che portino ad una soluzione. Nella pratica la classe di modelli matematici più utilizzata è quella della programmazione lineare.

1.2 Programmazione Lineare

Introduciamo in questo paragrafo alcuni concetti di programmazione lineare, rimandando alla bibliografia ([3], [11]) per coloro che volessero approfondire l'argomento. Il nostro obiettivo sarà quello di parlare dei problemi di programmazione lineare mista-intera e discuterne i principali metodi di risoluzione.

La programmazione lineare è una tecnica per l'ottimizzazione di una funzione obiettivo lineare, soggetta a vincoli lineari. La regione ammissibile è allora rappresentata da un politopo convesso definito dall'intersezione degli spazi dati dalle disuguaglianze lineari. Gli algoritmi di programmazione lineare hanno lo scopo di trovare un punto del politopo dove la funzione obiettivo ha il valore massimo o minimo, se esiste.

I campi di applicazione per i modelli lineari sono molti e vari e includono: trasporto, energia, telecomunicazioni.

Punto di forza di questa classe di problemi è l'esistenza di un algoritmo di risoluzione molto importante, chiamato Algoritmo del Simplesso. Questo deve la sua importanza al fatto che è un metodo di risoluzione esatto. Ciò significa che permette di trovare la miglior soluzione ammissibile, qualora questa esista, che risolve il problema studiato. Inoltre l'algoritmo è strutturato in modo tale che se il problema non ha alcuna soluzione ammissibile, è possibile saperlo con certezza.

1.3 Programmazione Lineare Intera e Mista-Intera

Un problema di programmazione lineare intera o mista-intera è un problema lineare nel quale tutte le variabili o una parte di esse sono vincolate ad assumere valori interi o binari. Generalmente i problemi di programmazione intera richiedono, nel caso peggiore, tempi computazionali molto elevati, esponenziale rispetto alla dimensione dei dati. Allo stesso tempo, sono di uso comune nelle applicazioni. Ad oggi esistono delle tecniche che permettono di risolverli in tempi ragionevoli nelle applicazioni più comuni. Per gli altri problemi, l'obiettivo è ricondurli a casi di studio già visti.

Consideriamo un problema di minimizzazione in programmazione lineare intera.

$$\begin{aligned} min \quad & c^T x \\ & Ax = d \\ & x \geq 0 \\ & x \quad intera \end{aligned} \tag{1.1}$$

Vista la difficoltà di tali problemi, si cercano di ottenere delle limitazioni superiori e inferiori del valore ottimo della funzione obiettivo. Per farlo dato il problema iniziale se ne considera il suo rilassamento continuo. Questo, si ottiene andando a modificare o eliminare i vincoli di interezza. Il rilassamento

continuo può essere risolto con le tecniche della programmazione lineare. La soluzione trovata del rilassamento continuo, se esiste, rappresenta un limite inferiore per la soluzione del problema iniziale, se fosse intera sarebbe allora anche soluzione del problema originario.

Vi sono casi particolari di programmazione lineare intera la cui soluzione del rilassamento continuo è sempre intera. Ciò avviene quando la matrice dei vincoli è totalmente unimodulare.

Definizione 1.1. Sia A matrice rettangolare, si dice totalmente unimodulare (TUM) se ogni sua sottomatrice quadrata ha determinante uguale a ± 1.

Proponiamo ora alcuni dei metodi esatti utilizzati per risolvere problemi di programmazione lineare intera. Questi possono essere estesi facilmente al caso dei problemi misti-interi.

1.3.1 Branch and Bound

È il metodo più utilizzato per risolvere problemi con vinvoli di interezza. L'idea è quella di andare a suddividere il problema iniziale P^0 in K sottoproblemi $P^1, ..., P^K$ che insieme rappresentino P^0.

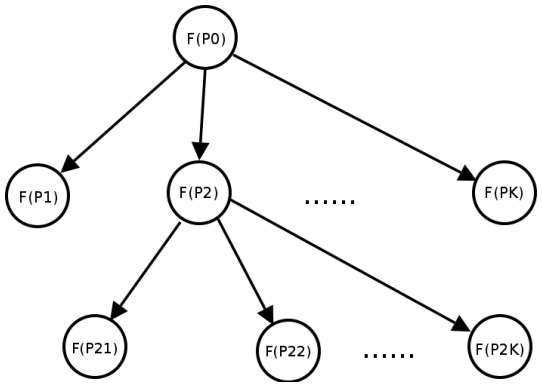

Figura 1.1: Suddivisione di un problema in sottoproblemi. Fonte: Wikipedia

Ciò può esser fatto, ad esempio, andando a suddividere la regione ammissibile del problema iniziale, partizionandola in modo tale che:

$$\bigcup_{i=1}^{K} F(P^i) = F(P^0),$$

$$F(P^i) \bigcap F(P^j) = \emptyset \qquad \forall P^i, P^j : i \neq j,$$

Questo processo solitamente chiamato branching (ramificazione) permette all'algoritmo di esplorare solamente i rami che potenzialmente possono avere una soluzione migliore.

Attraverso i rilassamenti continui dei sottoproblemi saremo in grado di trovare dei limiti inferiori per le loro soluzioni. A questo punto, trovata la soluzione ottimale di un sottoproblema attraverso il suo rilassamento continuo, se non risulta intera, andremo a suddividere ulteriormente in sottoproblemi, se la soluzione è intera, avremo una candidata soluzione per il problema iniziale. Saremo inoltre in grado di eliminare dall'albero decisionale tutti i rami che presentano un limite inferiore più alto della soluzione trovata. Questo processo ci permetterà di non visitare la totalità dei rami riducendo di molto i tempi di ricerca.

Appare chiaro come l'algoritmo dipenda molto da una stima efficiente dei limiti inferiori dei rami. Se questa non è disponibile, l'algoritmo degenera visitando tutti i rami.

1.3.2 Branch and Cut

Questo metodo comporta l'utilizzo dell'algoritmo di branch and bound combinato con piani di taglio, per restringere la regione ammissibile dei rilassamenti continui. Il limite inferiore del problema iniziale è fornito dal suo rilassamento continuo. Se la soluzione del rilassamento risulta non intera, l'algoritmo ricerca un vincolo violato da questa soluzione ma non violato da nessuna soluzione intera. Questo tipo di vincolo è chiamato piano di taglio. La disuguaglianza così aggiunta al rilassamento ci permetterà di ottenere una nuova soluzione ottimale, fornendo un migliore limite inferiore. Questa

operazione viene portata avanti iterativamente fino a quando o si trova una soluzione intera o diventa troppo costoso generare un altro piano di taglio. In quest'ultimo caso viene avviata la parte di branching del metodo. La ricerca dei piani di taglio continua sui sottoproblemi. Allora come nel caso precedente, un nodo dell'albero decisionale può non essere processato se presenta un limite inferiore più alto della soluzione trovata.

1.3.3 Branch and Price

Questo metodo è la combinazione del branch and bound con la generazione delle colonne. Viene utilizzato per risolvere problemi interi dove ci sono troppe variabili. Inizialmente solo una parte delle variabili viene considerata. Questo viene fatto al fine di ridurre i requisiti computazionali e di memoria. L'idea è che per problemi di grandi dimensioni la maggior parte delle variabili sarà uguale a 0, risultando irrilevanti per la risoluzione del problema. Per verificare l'ottimalità viene risolto un problema detto di pricing per trovare le variabili, quindi le colonne, in grado di ridurre la funzione obiettivo. Ad ogni nodo dell'albero decisionale delle variabili possono essere aggiunte. Generalmente i problemi di pricing possono essere difficili da risolvere, perciò si utilizzano metodi euristici. Se nessuna variabile può abbassare il valore della funzione obiettivo e la soluzione trovata non risulta intera, allora si attua un branching.

1.4 Google OR-TOOLS

Il modello è stato implementato in linguaggio **Python** e è stato risolto mediante l'utilizzo degli OR-Tools di Google. Questo, è un software open source per l'ottimizzazione, che cerca di trovare la soluzione migliore per un problema con un insieme molto vasto di soluzioni ammissibili. Trova impiego in numerosi casi come: problemi di programmazione dei percorsi, problemi di schedulazione e di caricamento dei mezzi.

Tali problemi solitamente presentano numerose soluzioni, gli OR-Tools sfruttano algoritmi all'avanguardia per restringere il set di ricerca. Sono presenti solutori per:

- Problemi di programmazione vincolata.
- Problemi di programmazione lineare, intera e mista-intera.

In quest'ultimo caso i solutori che troviamo sono: **GLOP** per problemi lineari come quelli di assegnamento e **SCIP** per la programmazione mista intera.

Nel nostro caso andremo a lavorare proprio con quest'ultimo per la risoluzione del modello. Il solutore **SCIP** utilizza algoritmi recenti, andando a suddividere il problema in sottoproblemi più piccoli (branching) che vengono poi risolti ricorsivamente attraverso metodi all'avanguardia di parallelizzazione e branch-cut and price. Rimandiamo alla bibliografia [6] per eventuali approfondimenti.

Capitolo 2

Elementi di Machine Learning

Il mondo delle reti neurali, e in modo più ampio del machine learning, negli ultimi anni è entrato prepotentemente nell'analisi dati e nel processo di decision making, non volendo questa tesi essere incentrata su tali argomenti, andremo ora a fare una spiegazione concettuale, senza addentrarci nel formalismo necessario, di alcuni elementi che ci permetteranno di comprendere meglio i risultati che vedremo più avanti.

Il machine learning è una branca dell'intelligenza artificiale che attraverso l'utilizzo di metodi statistici cerca di migliorare le performance di un algoritmo nell'identificare pattern nei dati. Infatti, per compiti semplici è possibile scrivere direttamente gli algoritmi che la macchina andrà poi ad eseguire, mentre per quelli più complessi ciò non è realizzabile. Allora risulta più efficace permettere alla macchina di sviluppare il proprio algoritmo. Si predispone la macchina a poter apprendere dai dati.

Gli algoritmi di machine learning costruiscono un modello probabilistico basandosi su dei dati, detti di addestramento, al fine di produrre previsioni sufficientemente accurate su dati che non ha mai affrontato.

Gli approcci di apprendimento solitamente sono classificati in tre ampie categorie:

- Apprendimento supervisionato: è una tecnica che mira a istruire il mo-

dello in modo da consentirgli di elaborare automaticamente previsioni sui valori di uscita di un sistema rispetto ad un input sulla base di una serie di esempi ideali, costituiti da coppie di input e di output, che gli vengono inizialmente forniti.

- Apprendimento non supervisionato: è una tecnica che consiste nel fornire al modello una serie di input che egli riclassificherà e organizzerà sulla base di caratteristiche comuni per cercare di effettuare ragionamenti e previsioni sugli input successivi. Al contrario dell'apprendimento supervisionato, durante l'apprendimento vengono forniti all'apprendista solo esempi non etichettati, in quanto le classi non sono note a priori ma devono essere apprese automaticamente.

- Apprendimento per rinforzo: è una tecnica che allena il modello usando un sistema di segnali di rinforzo (ricompensa), quindi il modello apprende interagendo con l'ambiente ricevendo una ricompensa se l'azione intrapresa risulta corretta o una penalità in caso contrario.

Un'altra classificazione dei modelli di machine learning si ha andando a considerare l'output del sistema:

- Classificazione: vengono trattati solitamente attraverso apprendimento supervisionato. Gli output sono divisi in classi e il sistema attraverso il modello deve assegnare gli input non ancora analizzati a una di queste. I modelli di machine learning più utilizzati in questo campo sono le reti neurali e alberi decisionali.

- Clustering: in cui un gli input vengono divisi in gruppi. Al contrario della classificazione, i gruppi non sono noti a priori sarà quindi il modello stesso a generarli. Questo rapprsenta tipicamente un modello non supervisionato.

Nel prossimo paragrafo andremo da analizzare una particolare sottoclasse del machine learning con la quale abbiamo lavorato nel corso dello sviluppo del modello. Rimandiamo alla letteratura per eventuali approfondimenti.

2.1 Reti neurali artificiali

Una rete neurale artificiale (ANN) è un modello computazionale, ispirato alla rete neurale biologica umana, utilizzato per tentare di risolvere problemi ingegneristici di intelligenza artificiale.

L'idea è quella di simulare cellule cerebrali densamente interconnesse all'interno di un computer per far sì che il computer apprenda cose, riconosca schemi e prenda decisioni. In termini pratici le reti neurali sono strutture non-lineari di dati statistici organizzate come strumenti di modellazione. Esse possono essere utilizzate per simulare relazioni complesse tra ingressi e uscite che altre funzioni analitiche non riescono a rappresentare.

Una rete neurale artificiale riceve segnali esterni su uno strato di nodi d'ingresso (input nodes), ciascuno dei quali è collegato con numerosi nodi interni (hidden nodes), organizzati in più livelli. Ogni nodo elabora i segnali ricevuti e trasmette il risultato a nodi successivi. Più complesso è il problema da risolvere con la rete neurale artificiale, più strati sono necessari. Ogni strato della rete contiene un certo numero di neuroni artificiali specializzati.

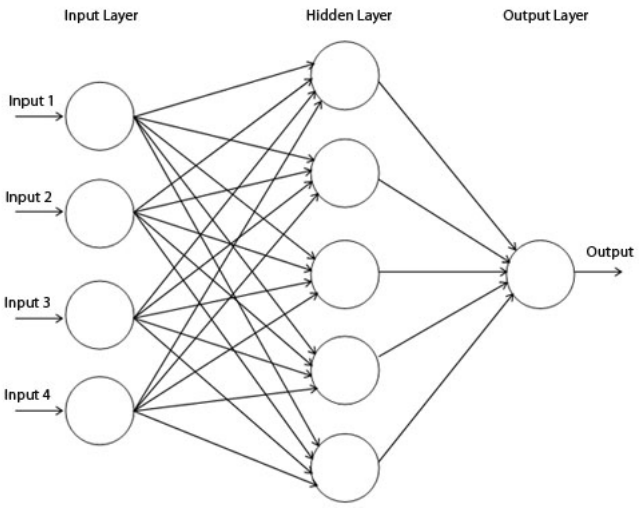

Figura 2.1: Schema rete neurale con uno strato di nodi interni

L'elaborazione delle informazioni nella rete neurale segue sempre la stessa procedura: le informazioni sotto forma di modelli o segnali sono trasferite ai neuroni dello strato di ingresso, dove sono elaborate. A ogni neurone è assegnato un peso, in modo che i neuroni ricevano un'importanza diversa. Il peso, insieme a una funzione di trasferimento, determina l'ingresso, dove quindi il neurone è inoltrato. Nella fase successiva una funzione di attivazione e un valore di soglia calcolano e ponderano il valore di uscita del neurone. A seconda della valutazione delle informazioni e della ponderazione, altri neuroni sono collegati e attivati in misura maggiore o minore. Per mezzo di questi processi è modellato un algoritmo che produce un risultato per ogni ingresso. A ogni addestramento, la ponderazione e quindi l'algoritmo è modificato in modo che la rete fornisca risultati sempre più precisi e migliori.

Un'importante distinzione tra le reti neurali è data dal tipo di connessioni che si hanno tra i nodi. Infatti le informazioni in una rete neurale possono propagarsi in due modi:

- Propagazione in avanti (feed-forward): il risultato di un nodo viene trasmesso ad un nodo dello strato successivo, si parla di propagazione semplice e aciclica.

- Propagazione all'indietro (feed-back), è una caratteristica delle reti più, complesse in tal caso il risultato viene trasmesso ad un nodo dello strato precedente, andando a generare dei cicli.

Le reti neurali per come sono costruite lavorano in parallelo e sono quindi in grado di trattare molti dati. Se alcune unità del sistema dovessero funzionare male, la rete nel suo complesso avrebbe delle riduzioni di prestazioni ma difficilmente andrebbe incontro ad un blocco del sistema. I software di ultima generazione dedicati alle reti neurali richiedono comunque buone conoscenze statistiche; il grado di apparente utilizzabilità immediata non deve trarre in inganno, pur permettendo all'utente di effettuare subito previsioni o classificazioni, seppure con i limiti del caso. Da un punto di vista industriale,

risultano efficaci quando si dispone di dati storici che possono essere trattati con gli algoritmi neurali. Ciò è di interesse per la produzione perché permette di estrarre dati e modelli senza effettuare ulteriori prove e sperimentazioni.

I modelli prodotti dalle reti neurali, anche se molto efficienti, non sono spiegabili in linguaggio simbolico umano: i risultati vanno accettati "così come sono", da cui anche la definizione inglese delle reti neurali come "black box": in altre parole, a differenza di un sistema algoritmico, dove si può esaminare passo-passo il percorso che dall'input genera l'output, una rete neurale è in grado di generare un risultato valido, o comunque con una alta probabilità di essere accettabile, ma non è possibile spiegare come e perché tale risultato sia stato generato. Come per qualsiasi algoritmo di modellazione, anche le reti neurali sono efficienti solo se le variabili predittive sono scelte con cura. Necessitano di una fase di addestramento del sistema che fissi i pesi dei singoli neuroni e questa fase può richiedere molto tempo, se il numero dei dati e delle variabili analizzate è molto grande. Non esistono teoremi o modelli che permettano di definire la rete ottima, quindi la riuscita di una rete dipende molto dall'esperienza del creatore.

2.2 Reti neurali ricorrenti

Diamo ora una definizione concettuale di una particolare sottoclasse di rete neurale ciclica detta ricorrente in cui una memoria degli input precedenti persiste nello stato interno della rete per essere utilizzati per gli input successivi.

Una rete neurale ricorrente (Recurrent Neural Network, RNN) è una classe di rete neurale artificiale che include neuroni collegati tra loro in un loop. Tipicamente i valori di uscita di uno strato di un livello superiore sono utilizzati in ingresso di uno strato di livello inferiore. Quest'interconnessione tra strati permette l'utilizzo di uno degli strati come memoria di stato, e consente, fornendo in ingresso una sequenza temporale di valori, di modellarne un comportamento dinamico temporale dipendente dalle informazioni ricevute

agli istanti di tempo precedenti. Le reti neurali ricorrenti permettono quindi di riconoscere patterns in una sequenza di dati.

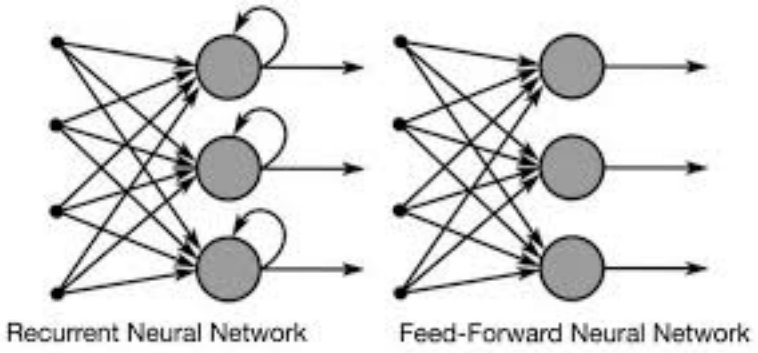

Figura 2.2: Esempio di rete neurale ricorrente e feed-forward. Fonte: [5]

La presenza di uno strato di memoria che persiste dall'input precedente, e influisce sugli output successivi rende le reti neurali ricorrenti un ottimo strumento per la predizione di serie di dati temporali. Gli algoritmi ricorrenti trovano largo utilizzo per il riconoscimento di testi o proteine e per l'analisi di dati di vendite o finanziari in cui l'aspetto temporale assume un ruolo fondamentale.

Punto di forza delle reti ricorrenti è quello di imparare automaticamente quali informazioni mantenere e quali ignorare, ciò gli permette di lavorare in modo ottimale con dati di vario tipo. Lo svantaggio è rappresentato dal fatto che non è facile costruire reti in grado di salvare informazioni per un lungo periodo, solitamente presentano una memoria a breve termine. Questo porta ad un problema che in letteratura viene chiamato "Scomparsa del Gradiente" (Vanishing Gradient"). Il valore del gradiente diviene troppo piccolo e l'algoritmo smette di apprendere o in ogni caso il processo può risultare molto lungo in caso di dati temporali estesi e/o di tipo multidimensionale. Per ovviare a tale limite, si utilizzano reti neurali dette Long Short-Term Memory (LSTM), che approfondiremo nel prossimo paragrafo.

2.3 Long Short-Term Memory

Si possono riconoscere due tipi di dipendenze: quelle a breve termine, che definiscono relazioni con il recente passato e quelle a lungo termine, che definiscono relazioni tra dati distanti nel tempo. Nel caso delle reti ricorrenti standard, la finestra osservabile è piuttosto limitata, in quanto più si torna indietro nella sequenza e minore è l'importanza che i valori appresi hanno sulla previsione corrente. Chiaramente questo impedisce al modello di apprendere e lo rende inefficace per set di dati estesi nel tempo. Determinate correlazioni tra i dati possono essere spiegate solo a lungo termine, usando solo il passato recente il modello potrebbe non essere in grado di riconoscere un certo schema.

A metà degli anni '90, venne proposta dai ricercatore tedeschi Sepp Hochreiter e Juergen SchmidHuber una variante di rete ricorrente in grado di apprendere dipendenze tra dati anche molto distanti nel tempo. Le reti Long Short-Term Meory (LSTM) consentono di ricordare gli input per un lungo periodo di tempo. Questo perché contengono le informazioni in una memoria, proprio come quella di un computer. Quindi l'LSTM può scrivere, leggere e cancellare le informazioni dalla sua memoria.

Questa cella di memoria autonomamente decide se conservare o eliminare l'informazione, in base all'importanza che gli attribuisce. L'importanza viene stabilita attraverso dei pesi che vengono anch'essi appresi dall'algoritmo. Ciò implica che quest'ultimo imparerà nel tempo ciò che è importante o meno. Sostanzialmente un LSTM presenta tre tipi di celle: input, forget, output. Attraverso di queste l'algoritmo determina se consentire il nuovo input (input), eliminare le informazioni perché non importanti (forget) o permettere che influiscano sull'output del passo temporale corrente (output).

Il problema della scomparsa del gradiente viene risolto dalle reti LSTM mantenendo, con le celle di memoria a lungo termine, i passi del gradiente abbastanza ripidi, permettendo un allenamento relativamente breve e con una precisione elevata.

Per ulteriori approfondimenti di questi argomenti, trattati solo concet-

tualmente, si rimanda ai testi in materia citati nella bibliografia.

Capitolo 3

Data Harmonization

L'informazione in ambito industriale ha sempre avuto un ruolo di primo piano nella pianificazione e nel processo di decision making. Risulta quindi fondamentale capire il flusso che porta dai dati, di cui un'azienda dispone in grande quantità, all'informazione che essi contengono. Infatti non sempre i dati portano con loro nuova informazione e in alcuni casi le informazioni che si possono derivare da questi risultano errate.

I dati di cui solitamente si dispone in ambito industriale provengono da fonti eterogenee, ad esempio diversi clienti o diversi reparti, e come già detto vi è la possibilità che siano presenti errori all'interno del set. Quindi prima ancora di poterli utilizzare per le analisi, vi è la necessità di riorganizzarli e in caso di errori di adottare delle procedure per correggerli.

3.1 Extract, Transform, Load (ETL)

In ambito informatico si parla di ETL (Extract, Transform, Load), un processo di estrazione trasformazione e caricamento dei dati. Quest'ultimi provengono da numerose sorgenti e vi è la necessità di organizzarli coerentemente in un unico repository per le elaborazioni successive. L'ETL ci permette di disporre di dati di qualità e preparati, cioè con tutte le caratteristiche richieste dal sistema. I dati così ottenuti sono una versione normalizzata dei

dati iniziali, e presentano lo stesso livello di informazione. Una volta che i dati verranno organizzati e caricati in una forma appropriata, sarà possibile utilizzarli per procedure di analisi e valutazione finalizzate alla pianificazione.

Il processo di ETL risulta solitamente diviso in tre fasi:

- Estrazione: in cui i dati grezzi vengono estratti da varie origini, come database esistenti o comuni file di testo, e collocati in un repository, come una Data Warehouse o un Data Lake.
- Trasformazione: in questa fase i dati subiscono un processo di trasformazione al fine di rendere omogenei dati provenienti da sorgenti diverse, e inoltre importante renderli conformi alle richieste delle elaborazioni successive.
- Caricamento: i dati una volta estratti e trasformati vengono caricati in una nuova destinazione.

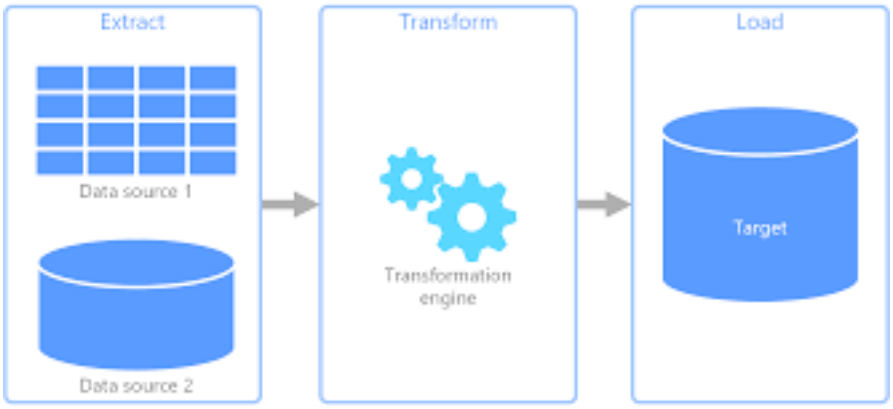

Figura 3.1: Schema riassuntivo ETL. Fonte: [10]

All'interno del processo di trasformazione i dati vengono sottoposti ad una fase di pulitura, al fine di migliorare la qualità di quest'ultimi. Alcuni degli errori tipici che si riscontrano sono:

- duplicazione di dati,
- mancanza di dati,

- inconsistenza tra i valori presenti.

Quindi si ha da parte dell'ETL una correzione e omogeneizzazione della moltitudine di dati provenienti dalle varie sorgenti. Questo processo viene chiamato Data Cleaning o pulitura dei dati. Rappresenta un procedimento fondamentale che precede l'effettiva estrazione di informazione utile dai dati.

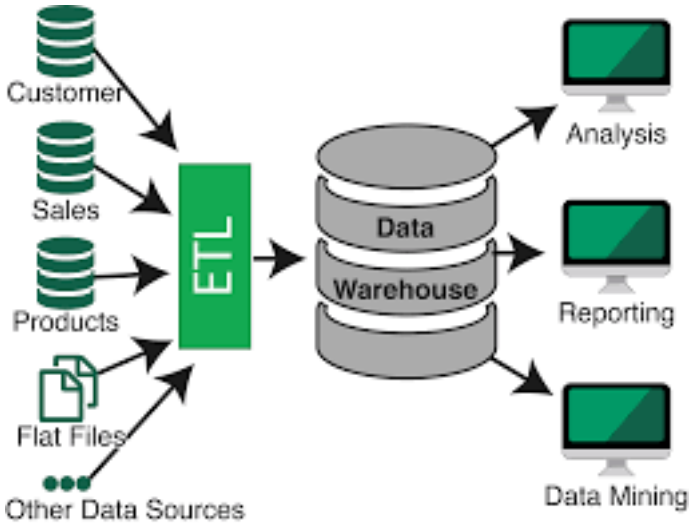

Figura 3.2: Schema del flusso di dati: dalle sorgenti, attraverso l'ETL fino alla fase di analisi e di Data Mining. Fonte: [1]

Negli ultimi anni nel processo di trasformazione dei dati si attua inoltre un processo di armonizzazione dei dati. Nel quale i dati vengono migliorati attraverso il ricorso a strumenti di apprendimento automatico. Tratteremo più approfonditamente la Data Harmonization nel prossimo paragrafo.

3.2 Armonizzazione dei dati

La Data Harmonization, di cui fa parte la Data Reconciliation, interpreta alcune delle caratteristiche dei dati, utilizzandole per trovare modifiche che né migliorino la qualità. In alcuni casi, tra cui quello che osserveremo

più avanti, i dati in possesso di un'azienda non contengono errori formali, come quelli elencati prima e perciò corretti nella fase di pulitura. Presentano però dei valori che seppur consistenti possono creare dei problemi per le fasi successive di training delle reti neurali, ad esempio dei valori riportati erroneamente. Quindi, un aspetto fondamentale della data harmonization è quello di combinare l'uso di dati sia da un punto di vista del business sia da un punto di vista IT (information technology). In questo modo si otterranno dati non solo formalmente corretti ma che presentano valori in grado di ben descrivere le dinamiche del sistema di cui fanno riferimento.

L'armonizzazione dei dati rappresenta uno snodo importante, la presenza di errori involontari possono portare a considerazioni e quindi decisioni errate. Comportando ad esempio una produzione maggiore del necessario o inferiore. Perciò, questo particolare processo viene applicato sopratutto in ambito aziendale, in particolare nel settore delle vendite e del marketing.

L'armonizzazione dei dati è una tecnica che mira a correggere gli errori di misurazione che sono dovuti al rumore, cioè errori casuali. L'assunzione principale è che non esistano errori sistematici nell'insieme delle misurazioni, poiché essi possono distorcere i risultati dell'armonizzazione e ridurne la robustezza.

Date n misurazioni y_i il problema della data harmonization, può essere espresso come un problema di ottimizzazione della forma:

$$\min_{x,y^*} \sum_{i=1}^{n} \left(\frac{y_i^* - y_i}{\sigma_i} \right)^2 \tag{3.1}$$

soggetto a:

$$F(x, y^*) = 0$$

$$y_{min} \leq y^* \leq y_{max}$$

$$x_{min} \leq x \leq x_{max},$$

dove y_i^* rappresenta il valore riconciliato della misura i-esima ($i = 1, ..., n$), y_i è il valore presente all'inizio del processo ($i = 1, ..., n$), x_j è la j-esima

variabile non misurata ($j = 1, ..., m$) e σ_i è la deviazione standard della i-esima misura ($i = 1, ..., n$), $F(x, y^*) = 0$ sono i vincoli di uguaglianza del processo e $x_{min}, x_{max}, y_{min}, y_{max}$ sono i limiti delle variabili misurate e non. Il termine nella funzione obiettivo, viene chiamato penalità della misura i. La funzione obiettivo sarà allora la minimizzazione della somma delle penalità. Quindi, il fine è minimizzare la correzione complessiva che è necessaria per soddisfare i vincoli del sistema. Inoltre, ogni termine dei minimi quadrati è pesato dalla deviazione standard della misura, in modo tale da normalizzare la quantità.

3.3 Stato dell'arte

Siamo pronti a questo punto a mostrare alcuni degli approcci di data harmonization o reconciliation. Daremo solo una panoramica di varie applicazioni in vari ambiti, al fine di osservare lo stato dell'arte di questa tecnologia. Rimandiamo alla bibliografia ([4], [2]) coloro che fossero interessati ad approfondire quanto stiamo per vedere.

Sistema DANCE

Solitamente per la pulizia dei dati si utilizzano regole di integrità e coerenza dei dati, per identificare gli errori e risolverli automaticamente. Queste soluzioni automatiche non possono garantire la precisione della riparazione. A tal fine si utilizzano esperti del dominio per determinare quali dovrebbero essere le modifiche da applicare al database. Però l'enorme volume dei database rende questa pratica manuale proibitiva. Quindi solitamente vengono attuati dei metodi al fine di ridurre lo spazio delle modifiche possibili, cercando di massimizzare il beneficio tratto dalla pulizia.

L'algoritmo DANCE trovata una violazione di un vincolo, identificherà i dati all'interno del database la cui modifica potrebbe portare al soddisfacimento del vincolo. Per ogni dato di questo insieme andrà a studiare i

potenziali effetti della modifica. Per determinare il dato x da modificare, terrà conto di tre parametri:

- quali dati non violerebbero più il vincolo con una modifica del dato x,
- il numero di aggiornamenti da dover fare a x,
- l'incertezza dei valori nel database.

Ciò serve a determinare un grafo i cui nodi sono i dati che violano il vincolo e i suoi archi rappresentano la probabilità che la modifica di un nodo influenzi quelli collegati. In questo modo sarà facile cercare di capire quali sono i nodi e quindi i dati che conviene modificare per primi.

Testato su numerosi set di dati si è dimostrato efficace e efficiente per la pulizia dei dati.

Ottimizzazione mista-intera per riconciliazione dati

I dati di cui solitamente dispongono gli impianti produttivi possono essere affetti da errori casuali e a volte da quelli che vengono chiamati "gross errors", errori cioè grossolani. La riconciliazione dei dati viene allora usata per fornire dati di migliore qualità. L'errore di misurazione è un tipo di gross error dovuto solitamente a mal calibrazioni o malfunzionamenti. Qualora dati corrotti da questo tipo di errore venissero riconciliati, tale errore di diffonderebbe su tutte le variabili andando a diminuire la qualità dei dati. Allora risulta importante rilevare questi tipi di errori e ridurre il loro impatto sul resto dei dati.

Sono stati sviluppati schemi ricorsivi in grado di rilevare la posizione di un errore, andando a comportare solitamente la sua eliminazione dal set di dati. Altre tecniche sono invece basate su modelli empirici, in grado di rilevare gli errori. Una volta effettuata l'eliminazione, si procede con la riconciliazione dei dati.

Si è quindi cercato di trovare una tecnica in grado di combinare la riconciliazione dei dati con il rilevamento dei gross error. Questo è stato fatto mediante l'utilizzo di un problema di ottimizzazione mista-intera.

Rimandiamo alla bibliografia per i dettagli di questa tecnica. Riportiamo solamente che il metodo proposto, nonostante computazionalmente risulti dispendiosa, porta ad avere prestazioni migliori, sopratutto quando il numero dei gross error risulta essere piccolo.

Capitolo 4

Kantar - Consulting Division

Per lo sviluppo di questa tesi ho collaborato con il team di Trade Promotion Optimization di Kantar, che si occupa della parte di ottimizzazione e mi ha messo a disposizione i dati su cui lavorare e gli strumenti necessari per condurre le analisi.

Il proposito di Kantar - Consulting Division è quello di aiutare i clienti a capire cosa è successo, il perché e come poter migliorare nel futuro. L'obiettivo è indurre una crescita dei ricavi per le aziende di beni di consumo, attraverso lo studio dei dati storici e l'utilizzo dell'intelligenza artificiale e dell'apprendimento automatico.

4.1 Trade Promotion Optimization

Attraverso strumenti di analisi predittiva, come LSTM, il team propone piani promozionali che aiutino il cliente ad incrementare il ritorno degli investimenti. Per fare ciò il team di TPO tenta di determinare le correlazioni tra le caratteristiche di una promozione, concetto che chiariremo più avanti, e l'aumento della domanda rispetto alla domanda di base del prodotto. Attraverso queste osservazioni sarà poi in grado di fornire previsioni precise della domanda per le campagne promozionali future di un determinato prodotto.

4.2 Struttura dati

I dati su cui abbiamo lavorato rappresentano le promozioni. Per chiarire alcune delle considerazioni che troveremo più avanti, cerchiamo di spiegare le caratteristiche principali di una promozione: la prima parte di una promozione definisce il cliente e il prodotto a cui è riferita, nella seconda troviamo informazioni di tipo temporale come l'anno, il mese e la settimana di una promozione e la terza parte contiene informazioni sulle caratteristiche che una promozione presenta. Il tipo di promozione viene identificato tramite un insieme di variabili, chiamate meccaniche. La meccanica potrebbe essere booleana, quindi solo presente o meno, come un grande evento, un display e così via, o categorico, se sono fornite diverse modalità di esecuzione, come un volantino che potrebbe mostrare il prodotto in una pagina interna o quella frontale. Alcuni di loro potrebbero essere continui, come la percentuale di sconto di sellout e vengono trattati come numeri.

Elenchiamo ora alcune delle variabili presenti nel set di dati, con le quali abbiamo lavorato, che ci permettono di identificare in modo univoco una promozione.

- IDACTION: codice identificativo di una promozione.
- CODPRODUCT: codice identificativo del prodotto in promozione.
- CODCUSTLEV2: codice identificativo del cliente che ha effettuato la promozione.
- SSOUT: anno e settimana in cui è iniziata la promozione.
- ESOUT: anno e settimana in cui è terminata la promozione.
- QTYBASELINE: quantità di prodotto venduta in assenza di promozione.
- QTYUPLIFT: quantità di prodotto venduta con certa promozione.

- SELLOUT_DISCOUNT: valore intero da 0 a 9, indica la percentuale di sconto applicata dal cliente sul prodotto.
- MULTIBUY: variabile booleana, indica se il prodotto è sotto una promozione di tipo 2×1 o 3×1.
- DISPLAY: variabile booleana, indica se la promozione nel punto vendita è visibile sui display.
- LEAFLET: valore intero da 0 a 2, indica se la promozione è inserita nel volantino del punto vendita e se sì in che posizione (FRONT o INSIDE).
- NEW_LAUNCH_CAMPAIGN: variabile booleana, indica l'attuazione di una nuova campagna promozionale nel punto vendita.
- BIG_EVENT: variabile booleana, indica la presenza di un grande evento nel punto vendita.

4.3 MOW (Missing Or Wrong)

All'interno del processo di analisi dei dati storici, questi vengono processati attraverso il MOW, che ha il compito di attivare alcuni importanti flag per l'analisi dei dati. Il MOW agisce su un database più esteso di quello delle sole promozioni, prende in esame l'intera serie temporale di una coppia prodotto-cliente, considerando tutte le settimane, in promozione e non. Diamo la seguente spiegazione di un valore presente nella tabella su cui lavora il MOW.

- QTYINV: rappresenta la quantità di prodotto venduta nella settimana.

Sostanzialmente analizzando le QTYINV di una coppia prodotto-cliente il MOW determina delle fasce per il valore delle vendite. In base ai valori di QTYINV di una settimana si andranno ad attivare uno o più flag.

- FLG_PROMO: viene attivato nel caso la settimana risulti in promozione, sia quindi presente un IDACTION, e il valore di QTYINV sia superiore ad una certa soglia.

- FLG_WRONG: può accadere che una promozione venga pianificata ma in seguito non venga applicata, ad esempio per problemi del punto vendita, in quel caso la settimana risulterà di promozione ma non si osserverà un aumento delle vendite, la QTYINV risulterà troppo bassa. Un caso come questo viene marchiato dal MOW, che attiverà il flag, ciò determinerà la modifica della settimana che passerà dall'essere in promozione al non esserlo.

- FLG_MISSING: identifica una promo mancante. Può accadere che una settimana in promozione non venga correttamente riportata nel database. In quel caso si avrà che le vendite presentano un picco senza un'effettiva promozione. Viene attivato il relativo flag e in questo caso la QTYINV, che presenta un picco, viene interpolata con le altre settimane non in promozione, in modo tale da abbassare il valore delle vendite.

4.4 LSTM nella realtà Kantar - Consulting

Introduciamo due concetti fondamentali per la valutazione dell'efficacia di una certa promozione.

- Baseline: si occupa di gestire la quantità di prodotto che si prevede di vendere nel caso in cui non sia attiva alcuna promo

- Uplift: rappresenta il surplus della quantità venduta rispetto al valore di baseline, dovuto alla presenza di una promozione. Può essere espresso come un rapporto o come una differenza tra la quantità venduta in promozione e quella venduta senza promozioni attive nello stesso periodo.

L'analisi congiunta della baseline e dell'uplift di una coppia prodotto-cliente, permette di trovare le meccaniche migliori per le promozioni future di un cliente. Sostanzialmente la rete LSTM sfrutterà una parte corposa della serie storica come set di dati di training in modo tale da poter calibrare i pesi e calcolare la baseline e l'uplift per il prodotto. Successivamente il modello sarà in grado di fornire una previsione dei pezzi venduti in una promozione in base alle meccaniche attive. L'accuratezza della previsione verrà valutata attraverso un set di validazione, anch'esso inerente al passato. Saremo in grado quindi di osservare se le quantità predette dal modello attraverso il forecast si avvicinano a quelle reali. L'accuratezza verrà analizzata mediante l'indice AVG_ERROR, che risulta essere una media pesata degli errori sulle singole coppie prodotto-cliente analizzate dal LSTM, più è basso più i valori predetti dal modello saranno aderenti a quelli reali.

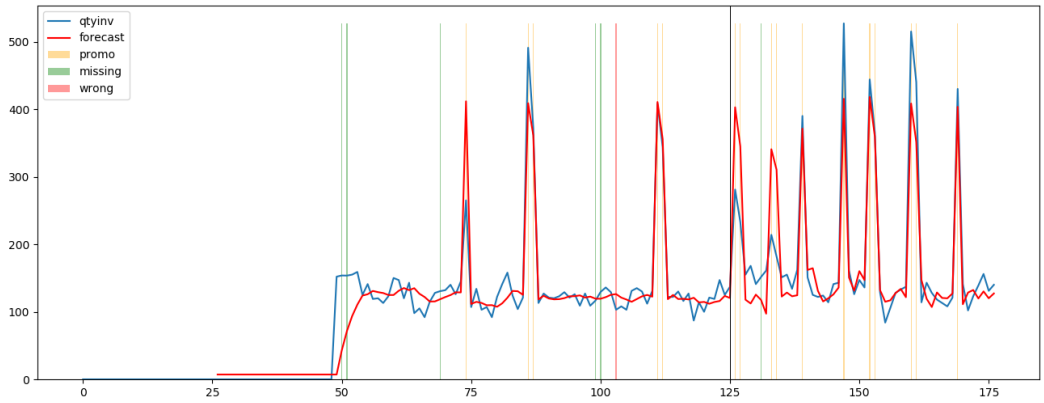

Figura 4.1: Esempio di serie storica: in rosso troviamo le quantità predette dal LSTM e in blu le quantità reali. Inoltre sono evidenziate le promozioni (in giallo), le FLG_MISSING (in verde) e le FLG_WRONG (in rosso).

Si comprende facilmente quanto sia importante riuscire ad abbassare l'AVG_ERROR, se la previsione risulta accurata, inserendo una determinata promozione con determinate meccaniche, il cliente sarà in grado di stimare in maniera ottimale quello che potrebbe essere il suo ritorno di investimento. Permettendogli di scegliere la tipologia di promozione e il periodo dell'anno più indicato per attuarla.

Capitolo 5

Il problema reale

Presentiamo in questo capitolo il problema affrontato nel corso del tirocinio presso Kantar - Consulting. Daremo prima spiegazione delle risorse utilizzate, poi chiariremo il problema fondamentale e gli approcci e metodi utilizzati preliminarmente e in conclusione riporteremo il modello finale.

5.1 I database

Il problema affrontato riguarda i valori all'interno di un set di dati contenente le promozioni di un cliente di Kantar - Consulting. Nel corso dello sviluppo del progetto lavoreremo con un due set di dati differenti:

- Il set $\mathcal{A}$ in cui ogni riga rappresenta una settimana. All'interno avremo sia settimane in promozione che non, entrambe avranno i codici identificativi di prodotto e cliente con relativa quantità venduta, ma solo le prime presenteranno il codice identificativo di una promozione e le varie meccaniche viste nel capitolo precedente.

- Il set $\mathcal{B}$ in cui ogni riga sarà una promozione. Allora per ognuna di esse avremo le variabili elencate nel capitolo precedente che indicheranno anche la durata della promozione.

Si potrebbe pensare che il set $\mathcal{B}$ sia un sottoinsieme del set $\mathcal{A}$, in realtà i due presentano strutture diverse, ciò porterà ad usare delle accortezze nello sviluppo del modello. Infatti in $\mathcal{B}$ ogni riga è una promozione, ciò vuol dire che ad esempio: una promozione che ha durata tre settimane, in $\mathcal{A}$ sarà indicata con tre righe ognuna con le opportune meccaniche e i valori di vendita, mentre in $\mathcal{B}$ la stessa promozione sarà indicata con una sola riga con le varie meccaniche e in cui il valore delle vendite sarà la somma dei valori delle righe in $\mathcal{A}$.

La necessità di lavorare con entrambi i database è dovuta al fatto che il processo aziendale procede sfruttando il database $\mathcal{A}$, ma il modello sviluppato è adattato alle caratteristiche del database $\mathcal{B}$. Questo implica anche che al termine del modello i risultati trovati andranno riportati sul primo database, per permettere il naturale procedimento dei dati all'interno dell'azienda.

Per il modello andremo a considerare solo le promozioni del database $\mathcal{B}$ che non sono state marchiate come *wrong* o *missing* dal MOW e che non presentano valori come 0 o *nan* nella colonna delle vendite. Dopo queste considerazioni il database $\mathcal{B}$ contiene 15033 promozioni.

$$\mathcal{B} = \{p_1, ..., p_{15033}\}$$

Per ognuna di queste andremo a costruire della variabili aggiuntive, oltre a quelle elencate nel capitolo precedente, che useremo nello sviluppo del modello:

- **nactive_mech**: è un numero intero che conta le meccaniche attive tra quelle in analisi.

- **active_mech**: è un set, in cui compaiono i numeri delle meccaniche attive, ad esempio: $(1, 2)$ indica che sulla promozione sono attive le meccaniche 1 e 2.

- **y_coeff**: è un valore reale, col quale andremo a confrontare le varie promo tra di loro e è dato da:

$$y\_coeff = \frac{QTYUPLIFT - QTYBASELINE}{(ESOUT - SSOUT + 1) \cdot QTYBASELINE}, \quad (5.1)$$

  rappresenta il rapporto tra l'aumento e la misura, andando a dividere per il numero di settimane in cui la promozione è stata attiva. Quindi, rappresenta un valore legato alla singola settimana e ci permette di confrontare promozioni che hanno anche durata molto diversa tra di loro. In seguito lo identificheremo per semplicità con y_{coeff}.

5.2 L'obiettivo

All'interno dei dati provenienti da un cliente è possibile ci siano degli errori, ad esempio dovuti a trascrizioni errate. Può quindi capitare che una quantità di merce venduta risulti errata, con un valore più alto o più basso di quello reale. All'interno del processo vi è uno strumento preposto all'identificazione di tali errori, il MOW, di cui abbiamo parlato nel precedente capitolo. Questo però confronta i valori delle vendite durante una promozione, la QTYUPLIFT, con la relativa QTYBASELINE, e in base ai valori presenti attiva i relativi flag.

L'obiettivo del modello è apportare delle modifiche alle quantità vendute confrontando le varie promozioni tra di loro. Idealmente, per allenare al meglio la rete neurale vorremmo avere che a promozioni migliori, corrispondono aumenti delle vendite più alti. In generale questo non è sempre vero, in quanto può capitare che per motivi dipendenti dal punto vendita una promozione ad esempio del 20% porti ad un aumento delle vendite maggiore di una promozione del 50% sullo steso prodotto. L'idea alla base della nostra scelta è che con dati che rispecchino una gerarchizzazione delle promozioni, che vedremo in seguito non essere così rigida, la rete LSTM allenata attraverso i nostri dati modificati porti ad un AVG_ERROR, quindi ad una accuratezza, migliore. Risulta anche chiaro che, qualora nel set di validazione vi

siano promozioni affette da errori con valori di vendite fuori norma, il modello andando a modificare tali promozioni riducendone l'errore migliorerà l'AVG_ERROR. Questo perché ovviamente LSTM non è in grado di predire valori affetti da errori casuali, quindi cercando di eliminare o ridurre tali errori, il forecast risulterà più accurato. Altra caratteristica importante del modello è che andremo a modificare il meno possibile i dati storici, così da non stravolgerli.

Specifichiamo che il nostro modello come anche l'analisi portata avanti dalla rete neurale lavora sulla coppia prodotto-cliente. Quindi, confronteremo promozioni sempre attinenti alla stessa coppia, questo tipo di analisi risulta meno complessa di un'analisi multi prodotto o multi cliente.

5.3 Modelli preliminari

Per arrivare al modello finale, di cui parleremo nel prossimo paragrafo, introduciamo i due modelli che inizialmente sono stati portati avanti parallelamente. Riporteremo solamente le caratteristiche principali e le motivazioni di questi due differenti approcci, cercando di giustificare la scelta del modello finale.

Entrambi si basano su una gerarchizzazione delle varie promozioni, che spiegheremo approfonditamente nel prossimo paragrafo, però mentre il primo modello modifica gli y_{coeff}, il secondo elimina le promozioni che presentano dei valori che non rispettano i vincoli di gerarchia.

Modello di data "modification"

Una volta definiti i vincoli del modello, questo modifica gli y_{coeff} delle promozioni che non verificano tali vincoli, attraverso l'utilizzo di due variabili reali positive. Queste, indicano di quanto il valore deve essere modificato verso l'alto o il basso. L'idea alla base di tale modello è che le modifiche da apportare al dataset fossero di piccola entità e polarizzate. Come funzione

obiettivo si andrà quindi a prendere la minimizzazione della somma di tutte le modifiche.

Modello di data "elimination"

Questo secondo modello, frutto di un differente approccio, va ad eliminare le promozioni che non verificano i vincoli, escludendole dal set di dati, attraverso l'utilizzo di variabili binarie, che indicavano se la promozione era da considerare o meno. La funzione obiettivo chiedeva di minimizzare il numero di promozioni eliminate. L'idea è che valori che risultino errati sopra una certa soglia anche se corretti andrebbero ad inficiare il processo di machine learning. In quanto, le modifiche avrebbero una certa entità e vi sarebbe il rischio di andare a modificare troppo il dataset storico. Perdendo quindi la prerogativa della data harmonization di non stravolgere i dati in possesso.

Entrambi i modelli sopra descritti sono stati testati solo preliminarmente, senza essere collegati all'effettiva rete LSTM. Abbiamo comunque fatto un analisi dei risultati osservando quali erano le modifiche apportate e le eliminazioni fatte. Queste analisi hanno evidenziato due fattori che hanno portato a quello che sarà il modello nella sua formulazione finale:

- nel caso del modello di data "modification" vi erano delle promozioni che venivano molto modificate, portando a stravolgere quasi completamente il dato di partenza, ciò è potenzialmente dovuto a valori non consistenti nella tabella iniziale,

- nel caso del modello di data "elimination" si è osservato che la percentuale delle promozioni eliminate risultava superiore al 10%, non partendo con dataset molto ricco, il rischio era quello di eliminare troppe promozioni andando comunque ad inficiare LSTM riducendo lo storico di training e il set di validazione.

5.4 Modello finale completo

Viste quanto riscontrato grazie ai modelli preliminari proposti, si è passati ad un modello in grado di risolvere le problematiche viste sopra. In cui si andavano a prendere elementi sia dell'uno che dell'altro.

L'idea di questo modello, in presenza di una promozione che non rispetti i vincoli, è di modificare i dati, fin quando la modifica non superi una certa soglia determinata dal utente, e qualora la modifica richiesta sia sopra tale soglia andare ad eliminare la promozione. Questo tipo di approccio ci permette da un lato di mantenere un set storico corposo, andando ad eliminare solo le promozioni affette da un errore importante, e dall'altro di apportare piccole modifiche in grado di migliorare l'accuratezza del LSTM.

Definiamo le varie componenti del modello:

- **Variabili:** per ogni promozione avremo due variabili reali positive e una variabile binaria. Le prime due indicano di quanto la y_{coeff} è stata modificata, verso l'alto o il basso. La terza indica se la promozione è attiva o è stata eliminata dal dataset. Quindi, dato $\mathcal{B} = \{p_1, ..., p_{15033}\}$ insieme delle promozioni avremo:

$$x_i, t_i \geq 0 \qquad i \in (1, ..., 15033), \tag{5.2}$$

$$z_i \in \{0, 1\} \qquad i \in (1, ..., 15033), \tag{5.3}$$

  dove le x_i diverse da zero indicano di quanto la y_{coeff} della promozione è stata modificata verso l'alto, le t_i di quanto è stata modificata verso il basso e le z_i valgono 1 se la promozione è stata eliminata e 0 se viene mantenuta.

- **Vincoli:** per definire i vincoli del modello, spieghiamo la gerarchia che abbiamo determinato tra le promozioni. Come già detto, confronteremo solo promozioni che agiscono sulla stessa coppia cliente-prodotto. Avremo quindi un sottoinsieme $\mathcal{C}_{j,k} \subset \mathcal{B}$ di tutte le promozioni che

presentano come cliente j e come prodotto k, cioè:

$$\mathcal{C}_{j,k} = \{p_i \mid CODCUSTLEV2 = j, CODPRODUCT = k\}. \quad (5.4)$$

Questi saranno gli insiemi per cui andremo a costruire i nostri vincoli confrontando le promozioni. Osserviamo che possa capitare che ci siano insiemi più popolati e insiemi poveri di promozioni, semplicemente nel primo caso si potrebbero avere più confronti e quindi sarà più facile correggere le quantità, nel secondo sarà più difficile per il modello determinare cosa fare.

All'interno di $\mathcal{C}_{j,k}$ due promozioni A e B risultano effettivamente confrontabili se siamo in uno dei seguenti casi:

- le due promozioni presentano le stesse meccaniche di tipo on/off, e le meccaniche a valori interi della promozione A sono entrambe minori o maggiori di quelle di B, nel primo caso A sarà inferiore a B nel secondo viceversa;
- le meccaniche di tipo on/off della promozione A sono un sottoinsieme di quelle di B, e le meccaniche a valori interi della prima promozione sono minori o uguali a quelli della seconda, nel primo caso la promozione A sarà inferiore a B, nell'altro sarà maggiore.

Ricordiamo che le meccaniche a valori on/off sono: MULTIBUY, DISPLAY, NEW_LAUNCH_CAMPAIGN, BIG_EVENT. Mentre quelle a valori interi: LEAFLET e SELLOUT_DISCOUNT.

Diamo un esempio pratico per comprendere come funziona questa gerarchia, consideriamo quattro promozioni p_1, p_2, p_3, $p_4 \in \mathcal{C}_{j,k}$:

PROMOZIONE	SELLOUT	LEAFLET	MECCANICHE ON/OFF
p_1	3	1	(MULTIBUY-DISPLAY)
p_2	2	1	(MULTYBUY)
p_3	4	0	()
p_4	2	2	(MULTYBUY)

Osservando le meccaniche attive nelle varie promozioni e i valori di SELLOUT e LEAFLET concludiamo che:

- p_1 è effettivamente confrontabile con p_2 a cui risulterà superiore, in quanto le meccaniche di p_2 risultano un sottoinsieme di quelle di p_1 e i valori di SELLOUT E LEAFLET sono entrambi maggiori uguali.
- p_1 non sarà confrontabile né con p_3 né con p_4, perché seppur vero che le meccaniche on/off sono un sottoinsieme di quelle di p_1, i valori delle altre meccaniche non sono entrambi minori o uguali a quelli di p_1.
- p_2 oltre ad essere confrontabile con p_1, lo sarà anche con p_4 in quanto le meccaniche attive sono le stesse e i valori delle meccaniche intere di p_2 sono minori o uguali a quelli di p_4. Quindi p_2 risulterà inferiore a p_4. Non sarà invece confrontabile con p_3 per lo steso motivo di p_1.

L'idea alla base è che vogliamo confrontare solo promozioni in cui appaia evidente che una sia superiore all'altra, non essendo in grado, con i dati in possesso, di determinare se ci sia una meccanica che influisca maggiormente sull'aumento delle vendite.

A questo punto, una volta chiarito quali sono le promozioni che andremo a confrontare siamo in grado di enunciare i vincoli che caratterizzano il nostro modello. Avremo due tipi di vincoli:

- **Vincolo di massima modifica:** per ogni promozione andremo ad imporre che la modifica del y_{coeff} non superi una certa percentuale, in modo tale da non stravolgere i dati iniziali. La percentuale di massima modifica, indicata con p, è un parametro scelto dall'utente. Il vincolo avrà la seguente forma:

$$x_i + t_i \leq p \cdot y_{coeffi} \qquad \forall i \in (1, ..., 15033) \qquad (5.5)$$

– **Vincolo di incompatibilità:** prese due promozioni confrontabili, quindi relative alla stessa coppia prodotto-cliente e per cui vale quanto detto sopra. Se la promozione A risulta superiore alla promozione B, ma il valore delle vendite, y_{coeff}, di A è minore o uguale a quello di B meno un certo parametro. Imponiamo nel modello che, se le promozioni risultano entrambe attive, quindi le variabili booleane sono entrambe uguali a 0, il valore riconciliato delle vendite di A sia maggiore o uguale a quello riconciliato di B meno lo stesso parametro di prima.

Ciò, si traduce nel seguente vincolo:

$$x_B - t_B - x_A + t_A - M(z_A + z_B) \leq y_{coeffA} - y_{coeffB} + \sigma \quad (5.6)$$

$$\forall (A, B) \in \{C_{j,k} \mid p_A > p_B \ , \ y_{coeffA} < y_{coeffB} - \sigma\}$$

Innanzitutto possiamo osservare che la parte con le variabili booleane serve ad imporre che il vincolo sia da verificare solo nel caso in cui entrambe le promozioni siano attive, cioè z_A e z_B sono uguali a 0. Ovviamente se una delle due promozioni non risulta attiva perché eliminata dal modello, il vincolo di incompatibilità non ha senso di esistere, allora essendo M un numero molto alto, il vincolo è automaticamente soddisfatto. Questo tipo di modellazione in letteratura viene chiamato, vincolo di tipo $bigM$ e permette anche di penalizzare la scelta del modello di eliminare una promozione.

Inoltre, possiamo osservare dal vincolo appena descritto che compare il termine σ, questo rappresenta il parametro di overlapping. Stiamo cioè permettendo ad una promozione gerarchicamente inferiore ad un'altra di avere un valore di y_{coeff} superiore all'altra ma non di molto. Questa scelta è determinata dalla volontà di non modificare troppo il dataset, siamo andati perciò con tale parametro a rilassare il vincolo di incompatibilità. Nei test portati avanti si è scelto come valore di overlapping la deviazione standard degli

y_{coeff} delle promozioni di una particolare coppia prodotto-cliente. Questa scelta in futuro potrebbe subire un cambiamento a seguito di operazioni di tuning sui parametri.

- **Funzione obiettivo:** quello che andremo a chiedere al modello è di minimizzare le modifiche sul set di dati. Sia in termini di modifiche ai valori y_{coeff}, sia in termini di eliminazioni apportate al dataset.

$$\min \sum_{i=1}^{15033} (x_i + t_i + z_i) \tag{5.7}$$

Il modello appena descritto ha risposto alle necessità evidenziate dai modelli preliminari, andando a ridurre la percentuale delle promozioni eliminate e apportando delle modifiche limitate ai dati.

A questo punto le modifiche devono essere applicate al set $\mathcal{A}$, attraverso le opportune trasformazioni.

Attraverso le variabili decisionali trovate dal modello, saremo in grado di determinare quali sono le promozioni da eliminare e quali quelle da modificare, verso l'alto o il basso. Spieghiamo ora come tali variabili vengono riportate sul set che poi verrà analizzato dal LSTM.

- Le promozioni eliminate dal modello, che ricordiamo avere durata anche superiore alla singola settimana, verranno interamente flaggate attraverso il FLG_WRONG in $\mathcal{A}$. Poiché non più promozioni i valori delle vendite, che possono presentare dei picchi molto elevati, verranno sostituiti interpolando con le altre settimane non in promozione. Ovviamente il flag viene esteso a tutte le settimane di promozione, così che se viene eliminata una promozione di quattro settimane, in $\mathcal{A}$ avremo quattro righe flaggate come wrong.

- Per le promozioni il cui valore di y_{coeff} è stato modificato, si calcolerà a quanto ammonta in percentuale, rispetto al valore iniziale, la modifica effettuata. Il valore percentuale così trovato, indicherà di quanto i

dati di vendita delle singole settimane in promozione di $\mathcal{A}$ debbano essere aumentati o diminuiti. Ad esempio se una una promozione di tre settimane ha subito un aumento del 20%, in $\mathcal{A}$ avremo che in ognuna delle tre righe relative a tale promozione i valori di vendita verranno aumentati del 20%. In questo modo andremo, in un certo senso, a distribuire l'aumento su tutte le settimane.

Una volta applicate queste trasformazioni ad $\mathcal{A}$, il dataset è pronto per essere processato dal LSTM, di cui osserveremo i risultati nel prossimo capitolo.

Capitolo 6

Risultati sperimentali

In questo capitolo andremo ad osservare i risultati raggiunti dal modello attraverso l'accuratezza del LSTM. Il fine ultimo della pulizia portata avanti sui dati è appunto quello di migliorare il set di training e di validazione su cui opera la rete neurale. Non andremo quindi ad osservare i valori della funzione obiettivo. I valori che osserveremo saranno l'AVG_ERROR, che come già detto è un indice che ci permette di capire quanto la rete neurale sia risultata accurata sull'intero set di validazione, e l'errore sulle singole coppie di quest'ultimo.

Cercheremo di capire in che modo le modifiche apportate dal modello abbiano influito sui risultati del LSTM, osservandone i vari grafici e interpretando le scelte del modello.

Come spiegato nel capitolo precedente, il modello, oltre ad andare a modificare le promozioni, potrebbe eliminarne. Questo all'interno del processo svolto dalla rete neurale, potrebbe in alcuni casi andare a diminuire il numero di coppie su cui il modello si addestra e di cui fornisce una stima. Infatti, per permettere una buona fase di training sulla coppia prodotto-cliente è necessario che questa disponga di un numero minimo di promozioni. Allora, nel caso in cui una coppia con poche promozioni veda eliminata una o più di esse dal modello, questa potrebbe essere estromessa dal processo del LSTM.

Questo non rappresenta un problema dal lato business, in quanto la coppia che sarà così eliminata è di interesse marginale per il cliente, in quanto già normalmente presenta pochi eventi di promozione. Precisiamo che, nel caso dell'AVG_ERROR andremo a confrontare quindi due database che potrebbero presentare delle differenze, in termini di numero di coppie presenti. A tal proposito, mostreremo anche l'errore medio non pesato, al contrario dell'AVG_ERROR, considerando solo le coppie presenti sia nel database ripulito dal modello che non.

Mostriamo l'andamento dell'AVG_ERROR al variare della percentuale di modifica ammessa nel modello:

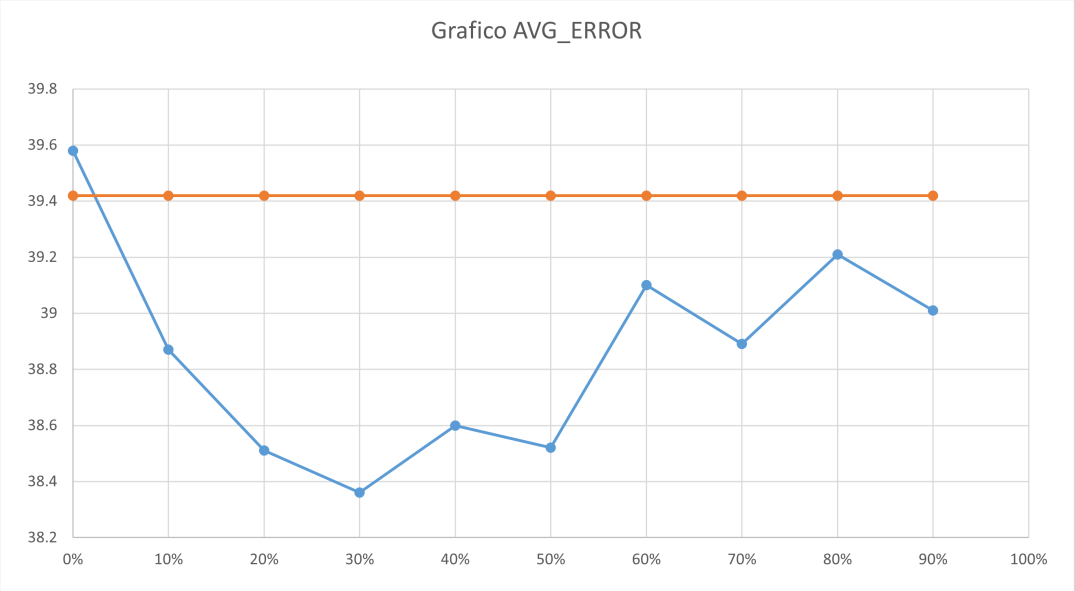

Figura 6.1: In arancione: LSTM senza modello. In blu: LSTM con applicazione del modello al variare delle percentuale di modifica ammessa.

Facciamo alcune osservazioni relative a tali risultati:

- Nel caso in cui la percentuale di modifica sia fissata allo 0%, ritorniamo al modello preliminare di Data Elimination, di cui abbiamo brevemente parlato nel capitolo precedente. Si osserva che tale scelta porta a

diminuire la precisione del LSTM, ciò è dovuto al fatto che andando ad eliminare le promozioni che non rispettano i vincoli di gerarchia, la rete neurale abbia meno dati su cui potersi allenare e le previsioni sul set di validazione risultano meno precise.

- Si osserva che, in qualunque caso, andare a modificare il set di dati, comporta un miglioramento nel LSTM. Risulta però importante constatare come permettere una maggiore modifica non comporti un miglioramento della precisione. Questo in quanto un set di dati molto modificato dal modello non risulta più aderente alla realtà del mercato che si vuole analizzare.

Questa prima analisi rappresenta un punto a favore del modello proposto, mostrando gli effetti positivi dell'applicazione di pulizia sul dataset. Inoltre, rende chiari anche i limiti di quest'ultimo. Sarà importante cercare di mediare l'azione del modello sul dataset con la necessità di conservare un numero consistente di dati e avere valori che rispecchino gli andamenti reali del mercato.

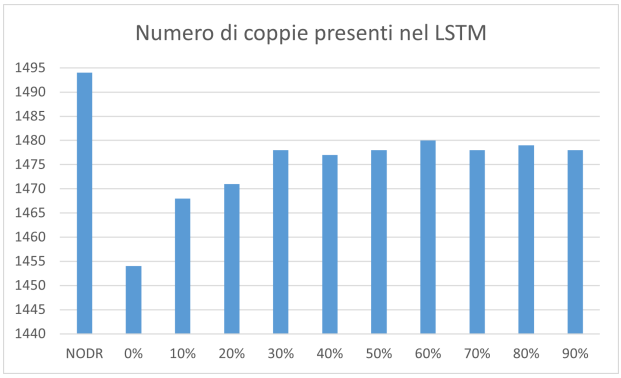

Figura 6.2: Numero di coppie presenti nel dataset al variare della percentuale di modifica ammessa.

Il modello, andando ad eliminare alcune delle promozioni, potrebbe ridurre il numero di coppie su cui LSTM opera. Ovviamente permettendo al modello modifiche di entità maggiore, il numero delle promozioni e quindi delle conseguenti coppie eliminate si riduce. Come però osserviamo dal

precedente grafico, il numero delle coppie eliminate si stabilizza intorno a 1478.

Questo potrebbe indicare che vi sia un set di promozioni che risulti in ogni caso sbagliato, quindi il modello, nonostante la percentuale di modifica aumenti, preferirà in ogni caso eliminare tali promozioni e di conseguenza le relative coppie. Ciò porta a pensare che all'interno del set di dati analizzato, vi siano delle promozioni con errori che non possono essere riconciliati con il modello in uso, ad esempio errori casuali sui valori di vendita.

Presentiamo ora il grafico in cui abbiamo confrontato l'errore medio non pesato del set senza data reconciliation e con, a parità di coppie presenti.

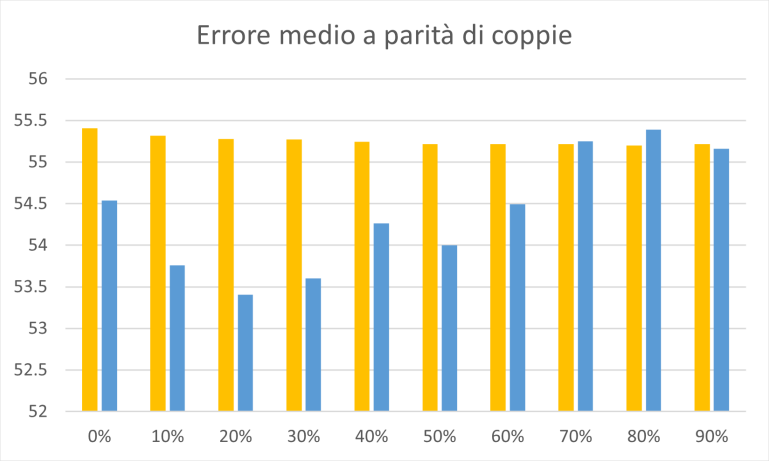

Figura 6.3: Errore medio non pesato a parità di coppie. In giallo: senza data reconciliation. In blu: con data reconciliation.

Si evince che l'andamento dei risultati visti per l'AVG_ERROR, si ritrovi anche in questo caso. Osserviamo come l'errore medio diminuisca quando la percentuale di modifica ammessa è intorno al $20\% - 30\%$, e aumenti all'aumentare della percentuale. Facendo questo confronto solamente sulle coppie presenti per entrambi i database, possiamo affermare che il miglioramento ottenuto non è semplicemente dovuto al fatto di aver eliminato delle coppie penalizzanti, cioè affette da un errore casuale elevato.

6.1 Risultati Data Reconciliation 30%

Andiamo ora ad analizzare nello specifico alcune delle coppie, prendendo i risultati ottenuti dal modello con percentuale di modifica ammessa pari al 30% in quanto risulta essere quello con l'AVG_ERROR più basso. Cercheremo, presentando alcuni esempi, di spiegare come ha agito il modello e perché.

Sulle singole coppie andremo ad osservare l'errore della precisione del LSTM sul set di validazione, vedremo quindi se il forecast prodotto si avvicina ai reali valori in nostro possesso.

Analisi coppie eliminate dal dataset del LSTM

Le coppie che, dopo la fase di pulitura del modello, vengono eliminate dal database in questo caso sono 16. Analizziamo due delle coppie eliminate, osservandone le serie storiche nel caso in cui il modello di data reconciliation non sia stato applicato.

- **Prima coppia.**

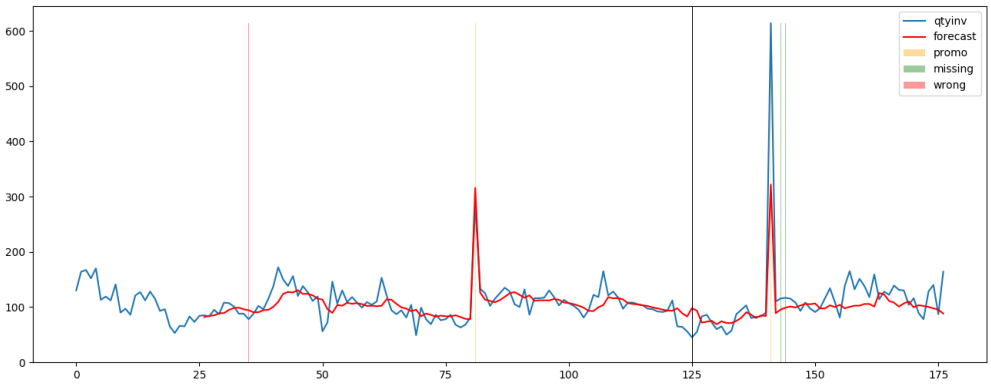

Figura 6.4: Serie storica della prima coppia senza data reconciliation.
Errore di forecast = 51.35%

Dal grafico si evince che la coppia presenta solamente due promozioni, la seconda di esse ha inoltre un picco elevato di vendite. Dall'analisi risul-

ta però che quest'ultima è gerarchicamente inferiore alla prima, presenta infatti dei valori inferiori nelle meccaniche di SELLOT_DISCOUNT e LEAFLET. Il modello, pertanto, sarà portato a riconciliare il dato delle vendite di tale promozione, cercando di abbassare il secondo picco. La modifica necessaria in questo caso risulta però superiore alla percentuale ammessa, comportando l'eliminazione della promozione che non rispetta il vincolo di gerarchia. Tale eliminazione comporta l'uscita della coppia dall'analisi del LSTM, in quanto il numero di promozioni non è sufficiente alla rete neurale per portare avanti un buon training.

- **Seconda coppia.**

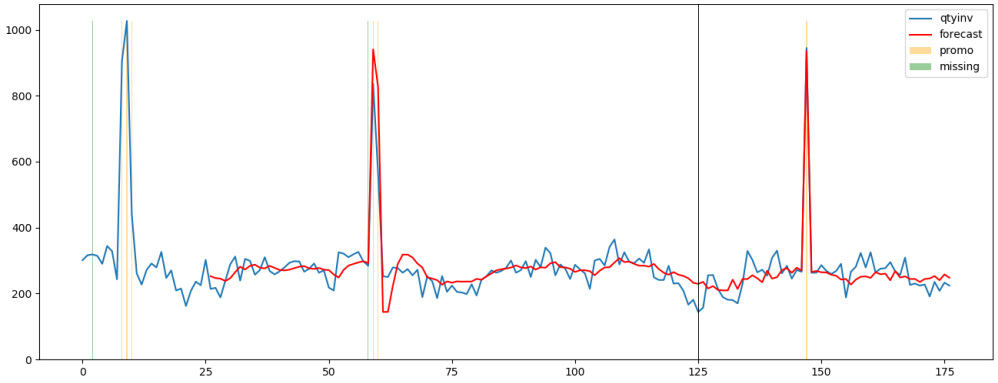

Figura 6.5: Serie storica della seconda coppia senza data reconciliation. Errore di forecast = 1.12%

Osserviamo subito che in questo caso il forecast prodotto dal LSTM sulla coppia risulta accurato, cioè approssima bene i vari picchi di vendite delle promozioni, quindi l'errore in questo caso risulta basso. Nonostante ciò la coppia viene eliminata dal set elaborato dal LSTM dopo la riconciliazione del modello. Analizzando le varie promozioni della coppia, si osserva come l'ultima promozione a livello temporale risulti gerarchicamente inferiore, ma con un valore di vendite maggiore rispetto alle altre. Il modello, in questo caso, è andato ad eliminare tale promozione dall'elenco.

Ciò indica che, per come è stato formulato e per la fase in cui è stato applicato, il modello, nel caso in cui rilevasse delle promozioni che non rispettino i vincoli di gerarchia, potrebbe eliminare o modificare promozioni anche nel caso in cui la rete neurale risulti essere in grado di ben approssimare l'andamento della coppia.

Analisi coppie con modifiche o eliminazioni

In questo paragrafo andremo ad osservare come l'errore di alcune coppie sia variato dopo l'applicazione del modello di data reconciliation. Andremo a confrontare i due grafici prodotti dal LSTM, nel caso in cui il database sia stato ripulito e non.

Mostreremo sia casi in cui l'errore di forecast è diminuito dopo la pulizia sia casi in cui è aumentato, cercando di darne spiegazione.

- **Prima coppia.** In questa prima coppia osserviamo come una delle promozioni sia stata eliminata dal modello. Questa, gerarchicamente inferiore alla promozione precedente, presentava un picco di vendite elevato. Questo potrebbe essere il risultato di qualche tipo di errore nella presa dei dati. Ovviamente un valore del genere andava ad inficiare l'AVG_ERROR, che per LSTM senza pulizia risulta molto elevato.

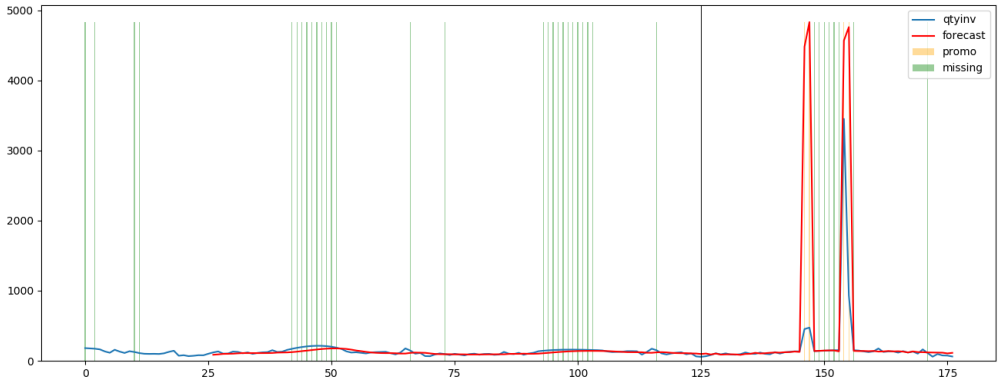

Figura 6.6: Serie storica della coppia senza applicazione del modello, errore di forecast = 632.89%.

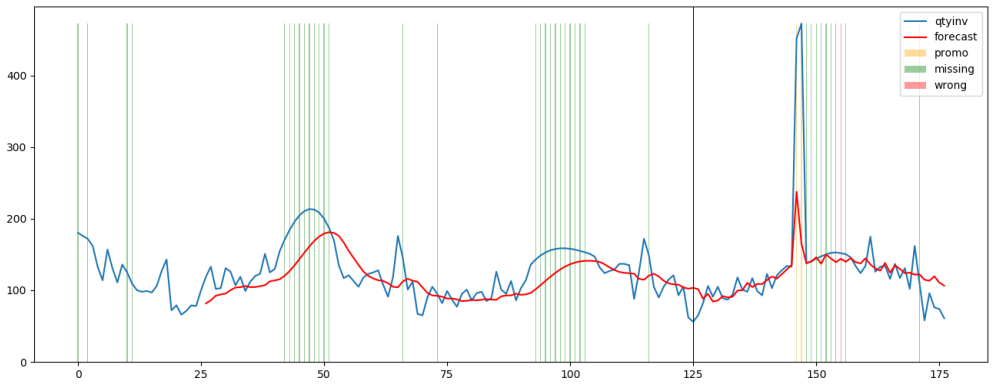

Figura 6.7: Serie storica con applicazione del modello, errore di forecast = 63.97%

Il modello in questo caso ha optato per l'eliminazione della promozione, il cui conseguente valore di vendite è stato ottenuto mediante interpolazione. La scelta del modello in questo caso ha migliorato di molto l'errore su tale coppia. Infatti, eliminando tale promozione il set di validazione risulta essere più coerente con il set di training della coppia. Di conseguenza il forecast prodotto dal LSTM risulta essere più accurato, osserviamo la diminuzione dell'AVG_ERRROR.

- **Seconda coppia.**

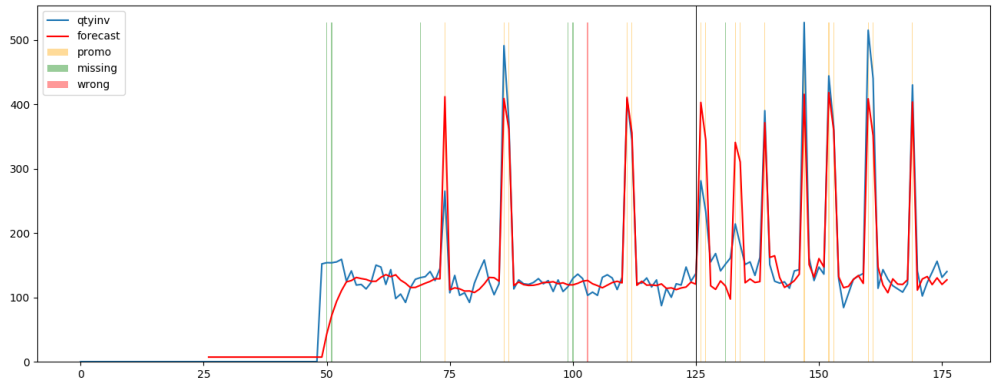

Figura 6.8: Serie storica della coppia senza applicazione del modello, errore di forecast = 27.41%.

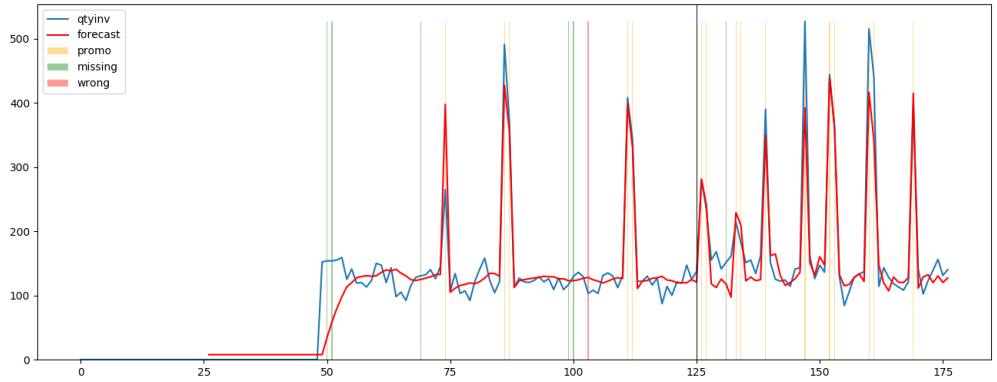

Figura 6.9: Serie storica con applicazione del modello, errore di forecast = 9.98%

In questo caso il modello non è andato ad eliminare una o più promozioni all'interno della coppia, ma ha apportato delle piccole modifiche ai valori di vendita dell'ultima promozione. Per la precisione analizzando i risultati del modello su tale coppia, i valori delle vendite dell'ultima promozione sono stati abbassati di circa l'8%. Tale coppia risulta essere un buon caso di studio in quanto sono presenti numerose promozioni, e questo permette al modello di avere più confronti tra di esse. Notiamo anche che la modifica apportata ha migliorato una coppia che presentava già un valore di AVG_ERROR non elevato.

Dalle prime due coppie proposte possiamo fare alcune osservazioni. L'eliminazione di una promozione risulta essere un buon modo per correggere le coppie che presentano un AVG_ERROR elevato. Quest'ultimo potrebbe di fatto essere dovuto alla presenza di errori casuali nei valori di vendita, e eliminando le promozioni che risultano errate andremo a migliorare il set di validazione con cui valuteremo LSTM. Mentre la modifica di alcune promozione ha apportato miglioramenti sopratutto nelle coppie che presentavano di per sé un AVG_ERROR poco elevato. Le modificazioni dei dati hanno di fatto permesso al LSTM di avere un set di training e di validazione più coerente con le dinamiche ideali del mercato senza però andare ad stravolgere lo storico dei dati provenienti dal cliente.

- **Terza coppia.**

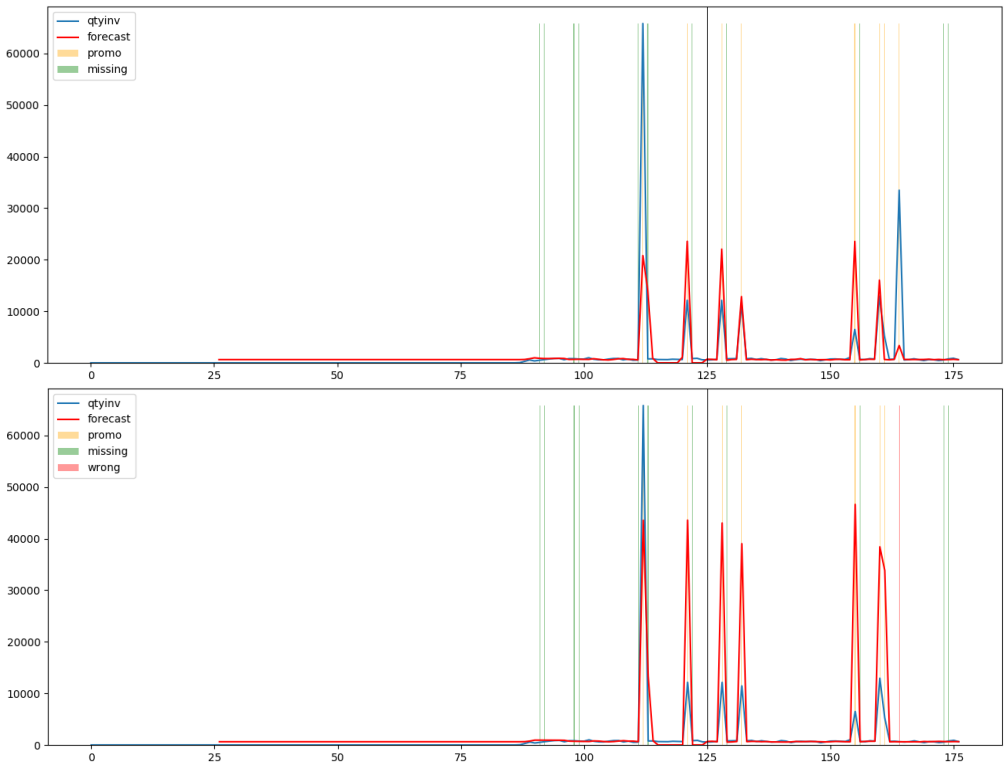

Figura 6.10: In alto: serie storica della coppia senza applicazione del modello, errore di forecast = 93.19%. In basso: serie storica con applicazione del modello, errore di forecast = 372.28%

In questo caso osserviamo come l'eliminazione di una promozione abbia di fatto aumentato l'AVG_ERROR della coppia. Il motivo di tale aumento è difficile da ricercare in quanto, come già detto, LSTM non è altro che una black box di cui siamo in grado di avere solo i risultati finali, pertanto daremo solo un un'ipotesi del perché ciò avvenga. La promozione che il modello è andato ad eliminare inizialmente non era ben approssimata dal LSTM, quindi probabilmente aveva dei valori di vendita molto elevati per le meccaniche che presentava. Dopo la sua eliminazione, quello che notiamo è che, i valori predetti per tutte le altre promozioni hanno degli aumenti importanti. Ciò potrebbe essere

dovuto a fattori interni alla meccanica del LSTM, infatti per predire i valori di una coppia, quest'ultimo, analizza i valori delle promozioni su quello stesso prodotto ma anche relativi ad altri clienti. Andando ad analizzare il cluster del prodotto, su cui LSTM opera, abbiamo osservato il seguente risultato: il cluster di quello specifico prodotto conta 12 diversi clienti, di questi solo 2 delle coppie prodotto-cliente vedono aumentare il proprio AVG_ERROR, tutte le altre vedono migliorare l'accuratezza.

Dall'analisi di tale coppia appare chiaro che il modello nelle sue prossime evoluzioni debba di necessità lavorare sul cluster del prodotto in promozione e non semplicemente con le coppie prodotto-cliente. In modo tale da avvicinarsi al tipo di operazioni effettuate dal LSTM. Questo dovrebbe permettere al modello di avere più confronti tra le varie promozioni, e quindi più vincoli, permettendo di migliorare i risultati. Il rischio potrebbe essere quello di avere così tanti vincoli da non avere più una soluzione.

Capitolo 7

Conclusioni

Il modello di ottimizzazione si proponeva di apportare delle modifiche ad un database storico di promozioni al fine di migliorare la precisione del LSTM. Sulla base dei risultati ottenuti il modello ha assolto il suo scopo. Portando ad un effettiva diminuzione dell'AVG_ERROR. Il modello rappresenta solo un punto di partenza da cui continuare ad investire e sul quale poter costruire uno strumento più efficiente e solido.

I prossimi passi per migliorare le prestazioni del modello possono svilupparsi in varie direzioni:

- ampliare la gamma delle meccaniche analizzate dal modello,
- calibrare, attraverso un tuning, il set di parametri che porti al migliore risultato,
- estendere l'analisi del modello dalla singola coppia prodotto-cliente, ad un'analisi multi prodotto e/o multi cliente.

Questo tipo di migliorie, visti i risultati fin qui ottenuti, pensiamo possano permettere un'analisi più completa, andando a rilevare ulteriori confronti possibili tra le promozioni e/o permettendo delle modifiche da parte del modello più precise. Tutto questo andrà poi a giovare al LSTM, che sarà in grado di rilevare e prevedere i picchi di vendita con una maggiore precisione.

Bibliografia

[1] Alger K.W., *Reducing the Need for ETL with MongoDB Charts*, Febbraio 2019, url: `https://www.mongodb.com/blog/post/reducing-the-need-for-etl-with-mongodb-charts`.

[2] Assadi A.; Milo T.; Novgorodov S., *Cleaning Data with Constraints and Experts(Technical Report)*, Conference: 2017 IEEE 33rd International Conference on Data Engineering (ICDE).

[3] Bruglieri M.; Colorni A., *Ricerca Operativa*, Zanichelli, Bologna, 2012.

[4] Capradossi V., *Long Short-Term Memory Neural Networks: an application on promotional actions revenue forecast*, Tesi Magistrale, Dipartimento di Scienze Statistiche, Università di Bologna, anno accademico 2018-2019.

[5] Donges N., *A guides to RNN: Understanding the basics of RNN and LSTM*, Giugno 2019, url: `https://builtin.com/data-science/recurrent-neural-networks-and-lstm`.

[6] Gamrath G.; Anderson D.; Bestuzheva K.; et al, *The SCIP Optimization Suite 7.0*, Marzo 2020, url: `http://www.optimization-online.org/DB_HTML/2020/03/7705.html`.

[7] Ionos Digital Guide, *Neural networks: cosa consentono le reti neurali artificiali?*, Marzo 2020, url: `https://www.ionos.it/digitalguide/online-marketing/marketing-sui-motori-di-ricerca/che-cose-una-rete-neurale/`.

[8] Nicholson C., *A Beginner's Guide to LSTMs and Recurrent Neural Networks*, 2018, url: `https://skymind.ai/wiki/lstm`.

[9] Soderstrom T.A.; Himmelblau D.M.; Edgar T.F., *A mixed integer optimization approch for simultaneous data reconciliation and identification of measurement bias*, Department of Chemical Engineering, The University of Texas at Austin, Austin, Aprile 2001, 869-876.

[10] Tejada Z., *ETL (Extract, Transform, and Load)*, Novembre 2019, url: `https://docs.microsoft.com/it-it/azure/architecture/data-guide/relational-data/etl`.

[11] Vigo D., *Lecture Notes on Integer Programming*, Università di Bologna, 2019.

[12] Wikipedia. *Data validation and reconciliation*, url: `https://en.wikipedia.org/wiki/Data_validation_and_reconciliation`.