



# Un insieme di Tecniche di Machine Learning per l'analisi delle Serie Storiche

Federico Ravenda<sup>1</sup>, Andrea Longobucco<sup>2</sup>, and Luca Necchi<sup>3</sup>

<sup>1</sup>Università degli Studi di Milano Bicocca, CdLM Clamses - STAT [Analisi Esplorativa, Feature Engineering, Modelling, Hypertuning, Entity Embedding]

<sup>2</sup>Università degli Studi di Milano Bicocca, CdLM Clamses - STAT [Analisi Esplorativa, Modelling, Stacking]

<sup>3</sup>Università degli Studi di Milano Bicocca, CdLM Clamses - SPI [Modelling]

## ABSTRACT

Nella presente trattazione sono stati studiati multipli approcci per lo studio delle serie storiche. In particolare sono state utilizzate diverse tecniche di ingegnerizzazione delle variabili e modelli di Machine Learning per valutare che impatto abbiano questi sui dati oggetto di studio.

Il dataset di riferimento proviene dalla competizione [Rossmann Store Sales](#) pubblicata 5 anni fa sulla piattaforma Kaggle. L'idea di concentrarsi su un *task time-series* nasce dalla volontà di studiare dati che si muovono nel tempo in un'ottica completamente diversa rispetto a quella che viene generalmente trattata durante un percorso di studi "standard" (utilizzando i tradizionali ARIMA), sfruttando le più recenti tecniche di Machine Learning per manipolare le variabili e creare modelli predittivi.

L'obiettivo della trattazione è proprio quello di esplorare diverse tecniche di apprendimento macchina e di fornire un overview sulle applicazioni di queste nell'ambito delle Serie Storiche.

Le analisi sono state svolte, principalmente, utilizzando il linguaggio **R**. In particolare, sono state utilizzate la librerie **h2o** [1] nella fase di *Feature Engineering* e **mlr** [2] per la fase di *Modelling* e *Stacking*. Infine si è utilizzato il linguaggio **Python** per implementare, con il pacchetto **Keras**, la sezione relativa all'*Entity Embedding*.

**Keywords:** Machine Learning, Time-Series, Sales, Feature Engineering, Ensemble

## CONTENTS

1	Introduzione	2	6	Un metodo Ensemble: Lo Stacking	9
2	Dataset – Quick Overview	2	7	Reti Neurali e Entity Embedding con Keras	10
3	Analisi Esplorativa	2	8	Conclusioni e Possibili Sviluppi	11
4	Preprocessing e Feature Engineering	5		References	11
4.1	Preprocessing	5			
4.2	Date-Related Features e Features Interactions	6			
4.3	Creazione di Nuove Features	6			
4.4	Target Encoding	7			
	Perché il Target Encoding? • Quali sono i Possibili Rischi?				
4.5	Anomaly detection: Isolation Forest	8			
5	Modelli ML per la Previsione	8			
5.1	Hypertuning	9			

## 1 INTRODUZIONE

La previsione delle serie temporali è sempre stata un'area di ricerca molto importante in molti domini perché diversi tipi di dati vengono stoccati come tali. Ad esempio, si possono trovare serie storiche in medicina, previsioni del tempo, biologia, gestione della catena di approvvigionamento, previsione dei prezzi delle azioni, ecc.

Data la crescente disponibilità di dati e potenza di calcolo negli ultimi anni, il Machine Learning è diventato una parte fondamentale della nuova generazione di modelli *Time Series Forecasting*, ottenendo ottimi risultati.

## 2 DATASET – QUICK OVERVIEW

Il dataset in esame si compone di:

- Un dataset di Training, costituito da 1'017'209 osservazioni
- Un dataset di Testing, costituito da 41'088 osservazioni

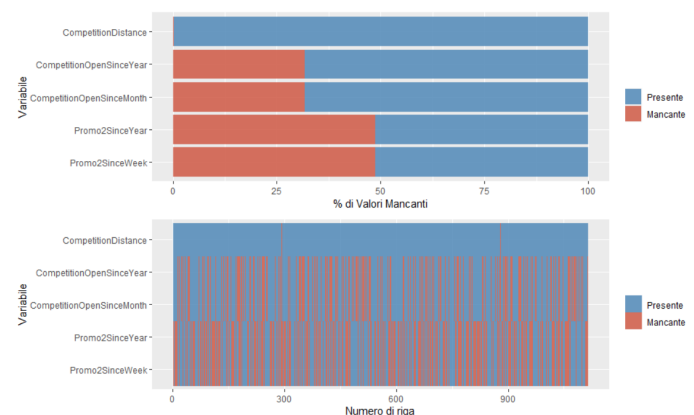
Le variabili del dataset sono le seguenti:

- **Id** - Valore unico che contraddistingue ciascuna osservazione.
- **Store** - Id che rappresenta univocamente ciascuno store.
- **Sales - Variabile Target**. Indica il fatturato di un determinato store in un dato giorno.
- **Customers** - Numero di clienti che un negozio ha avuto in una giornata.
- **Open** - Un indicatore che indica se lo store in questione è aperto: 0 = chiuso, 1 = aperto.
- **StateHoliday** - Può avere 4 categorie differenti, ognuna delle quali indica se il periodo considerato era di festa e, nel caso di risposta affermativa, anche di che tipo di festa si trattava. Le festività considerate sono Natale, Pasqua o festività pubbliche. *a = public holiday*, *b = Easter holiday*, *c = Christmas*, *0 = None*
- **SchoolHoliday** - Indica se lo Store, in quella precisa data, è stato influenzato dalla chiusura della scuola.
- **StoreType** - Indica la tipologia di store. Vi sono 4 differenti modelli di store: *a, b, c, d*
- **Assortment** - Descrive l'assortimento dello store: *a = basic*, *b = extra*, *c = extended*
- **CompetitionDistance** - Distanza espressa in metri dallo store competitor più vicino.

- **CompetitionOpenSince[Month/Year]** - Restituisce approssimativamente l'anno e il mese in cui il più vicino store competitor è stato aperto.
- **Promo** - Indica se un negozio sta offrendo una promozione in quel giorno specifico.
- **Promo2** - Promo2 è una promozione continua e consecutiva per alcuni negozi: se Promo2=0 allora il negozio non partecipa; se Promo2=1 allora il negozio partecipa.
- **Promo2Since[Year/Week]** - descrive l'anno e la settimana di calendario in cui il negozio ha iniziato a partecipare a Promo2.
- **PromoInterval** - descrive gli anni consecutivi in cui viene avviata Promo2, vengono citati i mesi in cui è iniziata di nuovo. Ad esempio: "Feb,May,Aug,Nov" significa che ogni volta ricomincia a febbraio, maggio, agosto, novembre di un dato anno per quel dato negozio.

## 3 ANALISI ESPLORATIVA

Si comincia l'analisi esplorativa controllando l'eventuale presenza di dati mancanti. Si può notare dalla **Figure 1** che sono 5 le variabili incomplete, nella fattispecie: "*CompetitionDistance*", "*Promo2SinceWeek*", "*Promo2SinceYear*", "*CompetitionOpenSinceMonth*" e "*CompetitionOpenSinceYear*".



**Figure 1.** Grafico relativo ai dati mancanti relativi agli store

Mentre nella prima tra queste variabili la presenza di valori mancanti è trascurabile, in quanto contiene solo lo 0.269% del totale; ciò non avviene invece nelle variabili "*Promo2SinceWeek*", "*Promo2SinceYear*" poiché contengono entrambe 48.8% di valori mancanti, così come questi non sono trascurabili nelle variabili "*CompetitionOpenSinceMonth*" e "*CompetitionOpenSinceYear*" che contengono il 31.7% dei missing values. Fatta esclusione per '*CompetitionDistance*', le altre 4 features in esame si riferiscono

alla presenza di competitors o alla presenza di una possibile campagna promozionale. Inoltre, nel momento in cui si prende in esame uno store che non ha nessun competitor nelle vicinanze, allora anche la distanza da questo o la sua data di apertura non vengono considerate. Lo stesso tipo di discorso può essere fatto per le variabili ‘Promo’. Infatti gli ‘NAs’ registrati per ogni riga della variabile “Promo2SinceWeek”, verranno registrati come tale anche nella variabile “Promo2SinceYear”; lo stesso ragionamento si può applicare per le variabili “CompetitionOpenSinceMonth” e “CompetitionOpenSinceYear”.

Poiché la variabile target è ‘Sales’ (ovvero il fatturato di un determinato store in un dato giorno), si è deciso di condizionare la risposta ai giorni della settimana al fine di potere analizzare quali siano i trend delle vendite durante ognuno di questi. Dai boxplot condizionati (Figure 2) risulta evidente una maggiore volatilità dei profitti ed un volume sensibilmente più alto di questi durante il lunedì; mettendo a confronto i boxplot tra loro risulta inoltre chiaro il decremento graduale dei profitti medi durante la settimana, fatta eccezione per il venerdì. E’ interessante notare che la domenica presenta profitti tendenti allo zero, questo si verifica poiché si considera

un’analisi grafica dei profitti espressi in centinaia di migliaia di euro. I profitti durante la domenica sono irrisori rispetto al resto della settimana poiché gli store sono chiusi (essendo chiusi tutti gli store non si possono registrare utili). Una conseguenza diretta della chiusura degli store si registra anche nei sabati; infatti molti store chiudono durante il weekend. Si può così dare una spiegazione al fatto che nei lunedì si registrino profitti tendenzialmente così alti: poiché in questo contesto ogni cliente non può, neanche volendo, fare acquisti nel weekend, in moltissimi saranno costretti a posticiparli e/o anticiparli. Infatti allo stesso modo, si può dare una spiegazione all’incremento dei profitti nei venerdì (che inizialmente sembrava apparentemente essere in controtendenza rispetto agli altri giorni della settimana). Si evidenzia la presenza di alcuni outlier da quasi tutti i boxplot, fatta eccezione per il lunedì e, ovviamente, la domenica (gli outlier risultano essere giorni in prossimità di festività come ad es. il Natale).

Nei grafici rimanenti si mettono a confronto le serie storiche dei profitti giornalieri condizionate ai diversi giorni della settimana. Sia sovrapponendole che analizzandole singolarmente risulta netto il divario tra il volume delle vendite nei lunedì rispetto a tutti gli altri giorni della settimana.



**Figure 2.** Vendite condizionate ai giorni della Settimana

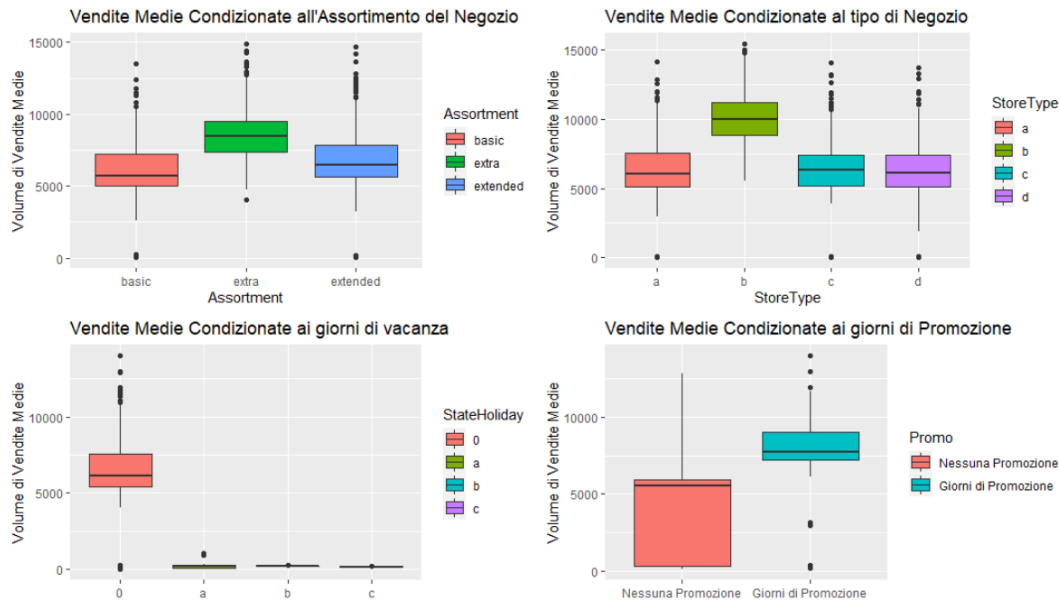
Si è voluto successivamente valutare l’evoluzione dei profitti in relazione ad altri fenomeni considerati all’interno del data-set (Figure 3). La scelta di condizionare dapprima i profitti rispetto alla variabile “Assortment” ha messo in evidenza che tendenzialmente gli store in cui il volume delle vendite medio è maggiore sono quelli che hanno un assorti-

mento di prodotti di livello ‘extra’ (rispetto agli store con assortimento ‘base’ ed ‘extended’). In entrambi i casi, comunque, gli store che presentano un livello di assortimento diverso da quello ‘base’ erano quelli che generavano più profitti. La variabilità nelle vendite negli store che hanno diverso assortimento è simile.

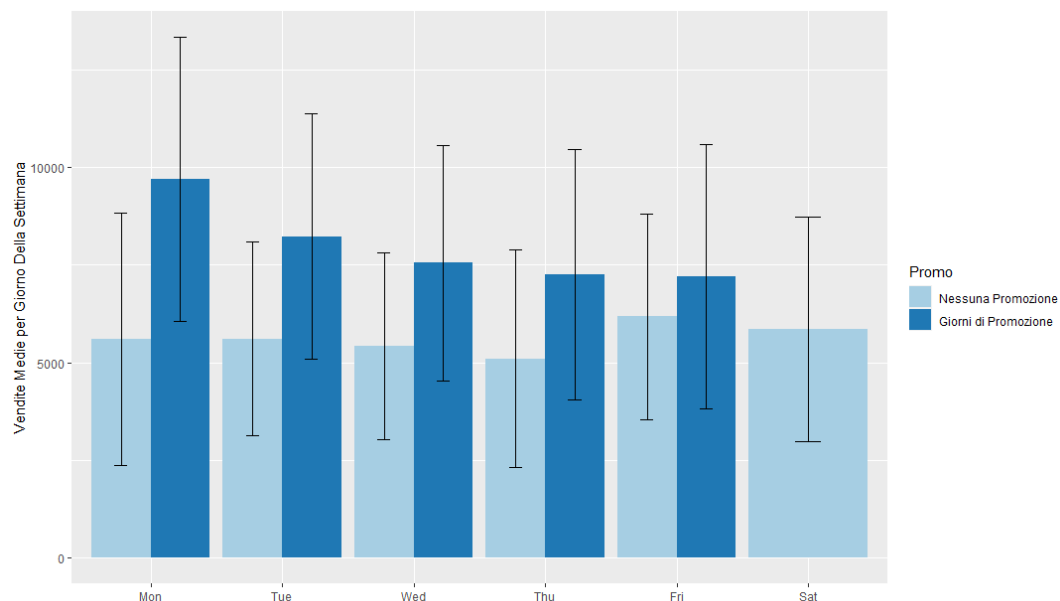
Successivamente si è deciso di adottare la stessa procedura per la variabile “StoreType”, che presenta quattro livelli e si riferisce a quattro ‘modelli’ di store differenti. Dai boxplot condizionati è evidente che le vendite medie negli store di tipo ‘b’ siano nettamente superiori rispetto alle vendite degli store appartenenti alle altre tre tipologie (che oltretutto sembrerebbero presentare distribuzioni abbastanza simili con valori di profitti medi molto simili tra loro).

L’analisi dei profitti rispetto alla variabile ‘StateHoliday’ conferma quanto detto in precedenza: le vendite durante i giorni festivi, ovvero nei giorni in cui la maggior parte degli store non lavora, sono così basse da essere quasi irrilevanti. Inoltre, per quanto riguarda le vendite compiute dai pochi store aperti, durante qualsiasi tipo di festività, si evince che

sono estremamente ridotte rispetto al periodo ordinario. Si è voluto poi analizzare come variano le vendite nei periodi in cui vengono applicate delle “Promo” nei diversi store; esclusa qualche rara eccezione (Lower Outlier), la presenza di “Promo” sembrerebbe incentivare così tanto i clienti nel compiere acquisti, da aumentare drasticamente le vendite medie rispetto ai periodi in cui queste non vengono applicate. Vista l’influenza della presenza di una “Promo” sui profitti di uno store, si è voluto mettere in evidenza attraverso l’ausilio di un diagramma a barre (Figure 4) come queste influenzino le vendite medie dei diversi giorni della settimana. Condizionatamente ad ognuno di questi, le vendite medie risultano sempre più alte quando è presente una promo. Si è inoltre voluto rappresentare anche le barre di errore.



**Figure 3.** Boxplot condizionati alle 4 diverse variabili categoriali



**Figure 4.** Grafico a barre condizionato alle Promo e ai giorni della Settimana

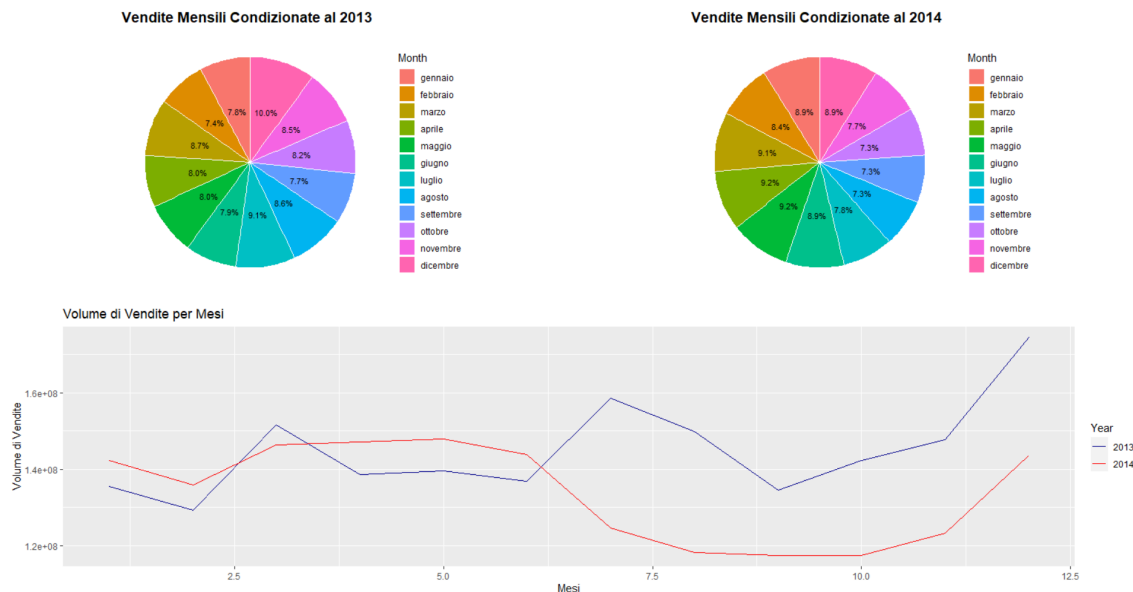
Si è deciso di analizzare la differenza nel volume delle vendite per quanto riguarda gli anni 2013 e 2014; è risultato infatti che dei 4.400.518.148 € di ricavi generati dalle vendite, ben 1.738.540.157 € sono stati ricavati nel 2013 e 1.607.505.352 € nel 2014. Nell'analisi esplorativa dedicata alle serie storiche dei ricavi mensili si è voluto escludere il 2015 poiché nel dataset di riferimento non sono presenti i dati relativi a tutti i mesi dell'anno considerato, per ogni store. Inoltre, si è prestata particolare attenzione nell'escludere gli store che nel corso del degli anni 2013 e 2014 hanno cessato la propria attività.

Risalta, dalle analisi fatte sulle serie storiche mensili condizionate all'anno preso in esame, che nel 2013 le vendite siano più volatili rispetto al 2014; queste inoltre sembrerebbero raggiungere dei picchi almeno una volta per ogni trimestre (marzo nel primo trimestre, maggio nel secondo,

luglio nel terzo e dicembre nel quarto). Il valore assoluto delle vendite nel 2013 è maggiore nel mese di dicembre, probabilmente poiché in concomitanza con le vacanze Natalizie. Inoltre si è notato che il periodo dei picchi coincida con la presenza di determinate promo.

Invece, nel primo semestre del 2014 l'andamento delle vendite risulta più costante; si è in presenza di un tracollo durante terzo trimestre dell'anno con una crescita esponenziale di queste durante il quarto trimestre (soprattutto, anche stavolta, nel mese di dicembre).

Nei grafici a torta è interessante valutare che la percentuale di vendite mensili condizionate all'anno in esame vari sensibilmente tra il 2013 e il 2014.



**Figure 5.** Percentuale delle vendite annuali condizionate ai mesi dell'anno

## 4 PREPROCESSING E FEATURE ENGINEERING

È risaputo che gli algoritmi di apprendimento utilizzano i dati di input per produrre previsioni. Molto spesso, tuttavia, i dati forniti potrebbero non essere sufficienti per creare un buon modello di machine learning. È qui che entra in gioco la potenza del *Feature Engineering*.

L'ingegnerizzazione delle variabili ha principalmente due obiettivi:

- Preparazione del set di dati di input corretto, compatibilmente con i requisiti dell'algoritmo di apprendimento automatico

- Migliorare le prestazioni dei modelli di Machine Learning

Di seguito, verranno approfondite le diverse tecniche utilizzate per creare nuova informazione dalle variabili a disposizione.

### 4.1 Preprocessing

Sono stati filtrati dal dataset di training tutti gli store che non erano presenti nel test set: così facendo si è notevolmente diminuito il numero di istanze presenti nel dataset (il numero di osservazioni è ora pari a 641'942).

Si è notato inoltre che la variabile Sales aveva una dis-

persione eccessivamente alta. Si è deciso di stabilizzare la varianza utilizzando una trasformazione logaritmica, la nuova variabile target è: **logSales**.

## 4.2 Date-Related Features e Features Interactions

Per modellare i dati delle serie storiche utilizzando l'apprendimento automatico, la funzione temporale deve essere suddivisa in sottocomponenti della serie storica.

Partendo, infatti, da una singola stringa contenente, per esempio, la data:

**'2015-07-31'**

si può ricavare una serie di informazioni, tra le quali:

- Giorno del Mese: (31)
- Mese dell'anno: (Luglio)
- Anno di riferimento: (2015)
- Quadrimestre di riferimento: (Secondo)
- Stagione dell'anno: (Estate)
- Giorno della settimana: (Venerdì)
- Giorno festivo o feriale: (Feriale)

Da una sola stringa di informazione si è riusciti ad estrarre 7 diverse variabili che si aggiungono al dataset. Alcune di queste variabili possono essere combinate al fine di creare ulteriori nuove features per incorporare la nostra conoscenza all'interno del modello. Si andranno a combinare:

- Giorno e Mese: se, per esempio, il giorno 31 Luglio è un giorno in cui i volumi delle vendite sono particolarmente elevati nel 2013 e nel 2014 allora anche nel 2015 l'algoritmo si aspetterà di mantenere questo trend.
- Mese e Anno

In modo tale che l'algoritmo possa creare una connessione tra le due variabili e generare una sorta di memoria del processo.

Di seguito, a titolo di esempio, il modo in cui sono state usate le interazioni:

Day	Month	Year	DayMonth	MonthYear
31	07	2014	31-07	07-2014

Ognuna di queste nuove funzionalità ha lo scopo di catturare alcune componenti delle serie temporali.

In questo modo, diventa possibile ottenere informazioni sulle caratteristiche temporali che influiscono sui dati, dato che la maggior parte degli algoritmi (soprattutto i più potenti, basati su alberi decisionali) non sono in grado di estrarre questa tipologia di dipendenza.

Alcuni dei metodi univariati comuni richiedono la stazionarietà nella serie. In altre parole, la serie ha media e varianza costanti. Questo è simile ai requisiti di normalità per l'utilizzo della modellazione di regressione. Applicando il feature Engineering ai dati e utilizzando un modello di apprendimento automatico, viene rimosso il requisito di stazionarietà nei dati. In generale, i metodi di machine learning riducono una serie di requisiti richiesti dai metodi tradizionali basati su serie storiche (ARIMA, HOLT-WINTERS, ...).

## 4.3 Creazione di Nuove Features

Soprattutto per quanto riguarda i *tree-based algorithm* La creazione di nuove variabili dai dati che si hanno a disposizione può migliorare drasticamente il potere predittivo dell'algoritmo. È qui che la conoscenza del dominio è estremamente importante: se si conoscono o si pensa di conoscere una qualche relazione, è utile includere variabili che descrivono tale relazione. Questo perché i metodi basati su alberi possono creare solo divisioni orizzontali o verticali (cioè ortogonali rispetto ai dati).

Sono state create tre nuove features: **nearbyChristmas**, **nearbyEaster**, **nearbyNewYear**.

Queste si riferiscono al numero di giorni che mancano rispettivamente a *Natale*, *Pasqua* e *Capodanno*. Sembra quindi sensato che nei giorni a ridosso delle suddette festività i Volumi di Vendita siano leggermente più alti.

Si è poi proseguito creando le suddette variabili:

- **Competitors**: *variabile dicotomica*. Assume valore 0 se, nella data considerata, non vi è alcun negozio concorrente nelle vicinanze.
- **timeComp**: *variabile numerica*. Nasce dall'esigenza di sostituire le variabili *CompetitionOpenSinceMonth*, *CompetitionOpenSinceYear* all'interno del dataset, ricche di Missing values. *timeComp* ci dice quanti mesi sono trascorsi dal momento in cui un è stato aperto un negozio concorrente nelle vicinanze.
- **hasPromoStarted**: *variabile dicotomica*. L'idea è quella di convertire la variabile *PromoInterval* in una nuova variabile che dice se nel mese corrente è iniziata una promo. Ci si aspetta che se nel mese cor-



rente è iniziata una promo le vendite saranno più alte.

Infine, è stato interrogato Google Trends (utilizzando la library `gtrendsR` [3]) direttamente dal compilatore R. Poichè i dati provengono da store localizzati in Germania, è stato chiesto alla funzione `gtrends` di ricercare il numero di volte in cui, nel motore di ricerca, in Germania, è stata cercata la parola 'rossmann' nel periodo 2012-12-29 2015-09-17. Al dataset è stato, quindi, aggiunto il valore che è

stato restituito dalla funzione in questione, ovvero il numero di 'hits' durante le settimane del periodo considerato.

## 4.4 Target Encoding

### 4.4.1 Perché il Target Encoding?

Nel dataset in oggetto, ci si è trovati di fronte a molte features categoriali, alcune delle quali hanno un numero di modalità estremamente alto, come si evince dal grafico 6

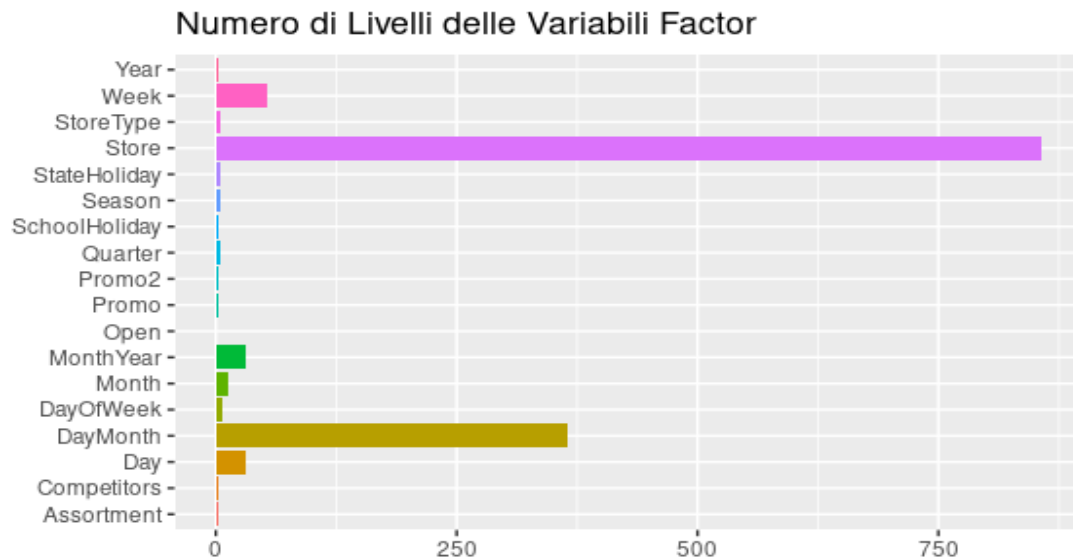


Figure 6. Numero di modalità per ciascuna delle variabili categoriali

Chiaramente con un così ampio numero di modalità si è deciso di scartare l'ipotesi di creare delle variabili dummy per ciascun livello di ciascuna variabile factor: inevitabilmente ci si sarebbe ritrovati a gestire un dataset troppo pesante che, con le risorse di computazionali che si avevano a disposizione, si sarebbe fatta molta fatica a stoccare. Aumentando il numero di variabili si sarebbe andati incontro alla problematica della *Curse of Dimensionality* [4]. Inoltre, i modelli parametrici come la regressione lineare, quando hanno in input troppe variabili, non sono in grado di stimare correttamente i coefficienti associati ai regressori, perdendo il loro potere predittivo.

Anche algoritmi basati su alberi come GBM soffrono l'elevata cardinalità delle variabili, infatti gli alberi di questi algoritmi hanno una profondità illimitata ma attraverso l'utilizzo del Mean Encoding si può compensare questo risultato [5].

Il target encoding è un approccio che proviene dal mondo Bayesiano: questo, infatti, usa l'informazione della variabile target per codificare le variabili categoriali. L'idea è quella di calcolare la media della variabile target per ciascuna modalità di ciascuna variabile categoriale e sostituire la categoria con il valore medio ottenuto.

Questo approccio non solo permette di risparmiare spazio di memoria e tempo computazionale nella fase di allenamento degli algoritmi, ma anche di creare un link tra la variabile target e la variabile categoriale (che, diversamente, con il One Hot Encoding, non si riuscirebbe a creare).

### 4.4.2 Quali sono i Possibili Rischi?

A volte applicare questo metodo può portare, inevitabilmente, all'*overfitting*: il modello apprende bene i dati di training, ma performa male nel set di validazione. Per questo motivo, spesso, si ricorre ad alcune tecniche di *regolarizzazione*.

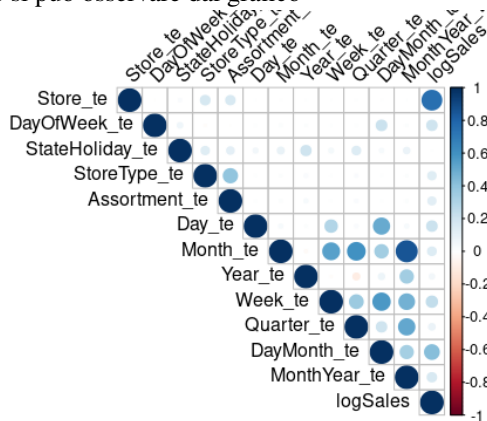
Quella utilizzata è stata la *CV Loop Regularization*, implementata tramite il pacchetto `h2o` di R. Tale tecnica è stata scelta poichè si tratta di un metodo molto intuitivo e robusto.

Le variabili che si è deciso di codificare tramite Target Encoding sono:

Store, Day, Month, DayOfWeek, StoreType, Assortment, DayMonth, MonthYear, Week,

Quarter, StateHoliday, Year, Season).

Come si può osservare dal grafico



Le nuove variabili create hanno, quasi tutte, una modesta correlazione con la variabile target logSales.

#### 4.5 Anomaly detection: Isolation Forest

È stata aggiunta, infine, un'ulteriore variabile che dà informazione agli algoritmi implementati circa il grado di anomalia di ciascuna osservazione.

Gli algoritmi di tipo *isolation forest*, implementati ancora una volta con il pacchetto R `h2o`, permettono di isolare ciascun punto del dataset sulla base della sua maggiore o minore distanza dagli altri punti. Gli outlier corrispondono in genere alle foglie che sono tagliate per prime dall'albero.

## 5 MODELLI ML PER LA PREVISIONE

A questo punto ci si concentrerà sulla modellizzazione previsiva al fine di cercare di risolvere nel modo migliore possibile il topic in esame. Saranno utilizzati diverse tipologie di oggetti finalizzati alla risoluzione del problema di regressione trattato e, a questo proposito, si cercherà di servirsi congiuntamente di strumenti di Statistical Modeling e di Machine Learning (Statistical Learning).

Come già anticipato in precedenza, è cura degli autori sottolineare l'ampio utilizzo della library di R `mlr`, poichè si è rivelata uno strumento di fondamentale importanza per il prosieguo del lavoro.

Si è deciso quindi di mettere in contrapposizione una serie di metodi di apprendimento parametrico e non parametrico, al fine di valutare quale potesse essere il *modus operandi* migliore per la risoluzione del topic.

Il pacchetto `mlr`, nei problemi di regressione, contiene di default alcune misure per la valutazione dei Learner ma, poiché il dataset in esame è stato preso da una competizione ormai terminata su Kaggle che utilizzava la misura **RMSPE** (*Root Mean Square Percentage Error*) come metro di valutazione, è sembrato opportuno 'customizzare' una nuova funzione nell'ecosistema `mlr` per la valutazione degli algoritmi previsivi basata su questa funzione di perdita.

I Learner utilizzati sono stati: **Random Forest**, il **modello di regressione multipla**, **Gradient Boosting Machine**, **Generalized Linear Model (GLM)** e le **Reti neurali**.

I learner considerati basati su algoritmi non parametrici, sono stati selezionati dalla libreria **h2o**. La scelta di utilizzare `h2o` ha reso possibile lavorare con un dataset piuttosto oneroso, sfruttando la potenza computazionale di Java per velocizzare la risposta degli algoritmi.

Successivamente si è passati alla valutazione degli algoritmi; dal training set sono stati esclusi gli ultimi due mesi di vendite che sono stati utilizzati nelle fasi di validazione e valutazione, considerando, nel validation set, lo stesso arco temporale che viene utilizzato nel test set. Inoltre, al fine di potere confrontare anche la loro efficienza in termini computazionali, si è voluto tenere memoria del tempo impiegato da ogni Learner a concludere la procedura di previsione e valutazione. Una prima valutazione degli algoritmi è stata fatta per modelli in cui non è stata condotta *features selection* (si sono considerate tutte le variabili a disposizione); per quanto riguarda gli iperparametri di ogni algoritmo, per il momento si è optato per l'utilizzo di quelli di default.

Di seguito i risultati:

Metodo ML	RMSPE	Runtime
Random Forest	0.1555640	254.258031
LM	0.1910201	2.051528
GLM	0.1909613	7.069968
GBM	0.1817633	27.886008
Reti Neurali	0.1874526	55.962340

Come si poteva immaginare, l'algoritmo che impiega più tempo per essere processato e valutato è *Random Forest* che però allo stesso tempo sembra essere, almeno per il momento, l'algoritmo più performante in termini di RMSPE. Per quanto riguarda le performance degli altri Learner, si ha una velocità media più alta e delle performance simili tra loro (fatta eccezione per le Reti Neurali che sembrerebbero performare meglio, pur essendo più onerose dal punto di vista computazionale).

A questo punto si è provato a valutare gli algoritmi allo stesso modo dopo aver fatto **Features Selection** (secondo i criteri espressi nel paragrafo precedente).

Per selezionare le variabili si è utilizzato *features importance* tramite Random Forest. La selezione di variabili ha permesso di ridurre considerevolmente il numero di variabili da considerare nel dataset. Le nuove variabili sono: **Store.te**, **DayMonth.te**, **Promo**, **hits**, **DayOfWeek.te**, **StoreType.te**, **is\_anomaly**, **CompetitionDistance**. I risultati sono i seguenti:



Metodo ML	RMSPE	Time
Random Forest	0.1634148	90.2010407
LM	0.1954590	0.8047318
GLM	0.1953809	4.3867278
GBM	0.1794535	23.2213511
Reti Neurali	0.1632955	43.3717141

Risulta interessante notare che il minore numero delle variabili abbia un buon impatto sulle dinamiche computazionali degli algoritmi diminuendo il tempo dedicato ad ognuno per essere processato e valutato (soprattutto in quelli più complessi). Per quanto riguarda le performance, il minore numero delle variabili influisce negativamente su Random Forest che addirittura perde quasi un punto percentuale rispetto alla valutazione precedente; un miglioramento sensibile avviene nelle valutazioni di Gradient Boosting Machine e nelle reti neurali (che addirittura guadagnano più di 2 punti percentuale rispetto alla valutazione fatta prima di Features Selection). Il Learner migliore sembrerebbe non essere più Random Forests ma le Reti Neurali.

## 5.1 Hypertuning

L'attenzione si è focalizzata in maniera particolare sugli algoritmi non parametrici *Random Forest* e *Gradient Boosting Machine*.

Per la fase di tuning si è utilizzato un criterio di ottimizzazione bayesiana sfruttando l'estensione **mlrMBO** della library **mlr**. Per quanto concerne Random Forest, si è voluto prestare attenzione all'ottimizzazione dei parametri discreti:

- **Ntrees**: il numero di alberi decisionali stimati per la foresta casuale; il tuning ha evidenziato come valore ottimale prima e dopo Features Selection `ntrees=70`.
- **Max\_depth**: la profondità massima di ogni albero decisionale; il tuning ha evidenziato come valore ottimale prima e dopo Features Selection `max_depth = 30`, `max_depth = 10`.
- **Min\_rows**: il numero minimo di osservazioni per ogni foglia di ogni albero decisionale; il tuning ha evidenziato come valore ottimale prima e dopo Features Selection `min_rows = 5`.

Per quanto riguarda invece Gradient Boosting Machine, i parametri sui quali è stato fatto tuning sono:

- **Ntrees**: il numero di alberi decisionali stimati; il tuning ha evidenziato come valore ottimale prima e dopo Features Selection `ntrees=120` e `ntrees=30`.
- **Max\_depth**: la profondità massima di ogni albero decisionale; il tuning ha evidenziato come valore ottimale prima e dopo Features Selection `max_depth=10` e `max_depth=15`.

- **Learn\_rate**: Questa opzione viene utilizzata per specificare la velocità con cui GBM apprende durante la creazione di un modello. I tassi di apprendimento più bassi sono generalmente migliori, ma sono necessari più alberi per raggiungere lo stesso livello di adattamento (Questo metodo aiuta ad evitare eventuale overfitting); il tuning ha evidenziato come valore ottimale prima e dopo Features Selection `learn_rate=0.29` e `learn_rate=0.23`.
- **Max\_abs\_leafnode\_pred**: Quando si crea un modello GBM, questa opzione riduce l'overfitting limitando il valore assoluto massimo di una previsione del nodo foglia; il tuning ha evidenziato come valore ottimale prima e dopo Features Selection `max_abs_leafnode_pred=10.1` e `max_abs_leafnode_pred=10.8`.

Dopo il tuning degli iperparametri, la valutazione finale sui modelli è la seguente:

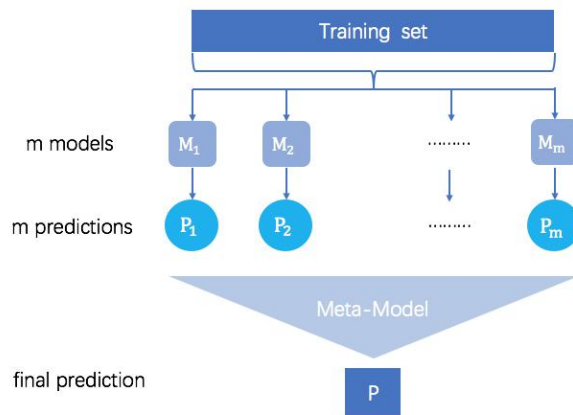
Metodo ML	Base RMSPE	FS RMSPE
Random Forest	0.1566754	0.1603070
GBM	0.1467940	0.1547073

## 6 UN METODO ENSEMBLE: LO STACKING

L'apprendimento d'insieme (Ensemble Learning), rappresenta un'importante costola del Machine Learning che ha come strumenti l'utilizzo di alcune metodologie volte all'ottimizzazione di un modello o di un algoritmo predittivo.

Il concetto cardine dell'apprendimento ensemble è quello di generare iterativamente più modelli basandosi su uno stesso set di dati sul quale vengono confutate più ipotesi per ogni iterazione dell'algoritmo. L'ensemble learning viene maggiormente utilizzato per la risoluzione di problematiche in cui sussistono relazioni non lineari nei dati; per questo motivo rappresenta una delle principali soluzioni per la ricerca di un metodo predittivo efficiente in ambito parametrico e, soprattutto, non parametrico.

A questo proposito si è voluto aprire un discorso a parte sulla metodologia di ensemble, generalmente, meno trattata in ambito accademico, al fine di dare agli scriventi la possibilità di fare chiarezza sulle capacità e sul funzionamento di tale strumento.



In sostanza, nello Stacking diversi weak learners vengono adattati in maniera indipendente l'uno con l'altro e un meta-modello (SuperLearner) viene addestrato su questi per prevedere gli output in base agli output restituiti dai modelli di base. I weak learners che presenteranno un errore complessivo minore avranno un peso maggiore nel computo del risultato finale. Lo stacking in genere, ma non sempre, produce prestazioni migliori di qualsiasi singolo modello stimato [6]. Inoltre uno dei principali svantaggi di questa metodologia sta nell'onerosità computazionale dell'algoritmo. Per quanto riguarda il suo utilizzo nel dataset "Rossman Store Sales", è fondamentale premettere che lo strumento è stato recuperato all'interno della library **mlr**, così come il resto dei modelli di machine learning utilizzati per la risoluzione del suddetto topic. Risulta interessante inoltre come le valutazioni sulle previsioni (RM-SPE) varino a seconda della tipologia di algoritmo utilizzato per stratificare i weak learners, ma anche al variare delle caratteristiche di questi ultimi. Nella fattispecie, per ogni combinazione di weak learners identificati si è proceduto con l'utilizzo di algoritmi Stacking:

- Voting Ensembles: l'importanza attribuita ad per weak learner è uguale; non esistono pesi e le previsioni vengono fatte semplicemente facendo una "media" di ciò che viene restituito da ogni learner.
- SuperLearner Ensemble: è stato identificato un SuperLearner affinché l'algoritmo funzioni come illustrato in precedenza; nel caso specifico in esame è stato identificato come SuperLearner un GLM poiché, secondo fonti antologiche, risulta quello più versatile e meglio generalizzabile nella risoluzione di problemi previsivi.

I risultati migliori restituiti dallo Stacking, in cui sono stati inseriti dei weak learners nei quali è stato fatto sia Hypertuning che Features Selection, sono stati i seguenti:

Tipologia	RMSPE
Voting Ensembles	0.154665
SuperLearner (GLM) Ensemble	0.1501540

Si può notare inoltre, come si poteva supporre già prima di valutare i due modelli, che le previsioni tra Voting Ensembles e SuperLearner Ensemble risultano migliori nel modello in cui è stato definito un SuperLearner.

## 7 RETI NEURALI E ENTITY EMBEDDING CON KERAS

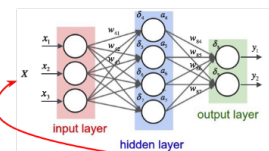
Si è voluto, infine, approfondire uno degli *hot-topic* del mondo del Deep Learning: si tratta del concetto di *Embedding*. Gli Embeddings, ormai ampiamente noti nel campo del *Text Mining*, sono delle rappresentazioni nello spazio ridotto in cui si possono tradurre dei vettori *High Dimensional*.

Gli Embeddings sono un'utilissima soluzione nella gestione delle variabili categoriali, poiché evitano molte delle insidie relative a un One Hot Encoding.

Senza avere la pretesa di approfondire eccessivamente gli aspetti teorici/geometrici di questo metodo ci si soffermerà su quella che è l'idea *core* alla base del metodo: Utilizzare arrays di dimensione diversa per rappresentare un insieme di variabili categoriali.

Questo offre 2 vantaggi:

- Si limita il numero di colonne del dataset.
- Gli Embeddings per natura raggruppano variabili intrinsecamente simili.



Sunday	0.4	-0.3	0.6	0.1
Monday	0.2	0.2	0.5	-0.3
Tuesday	0.1	-1.0	1.3	0.9
Wednesday	-0.6	0.5	1.2	0.7
Thursday	0.9	0.2	-0.1	0.6
Friday	0.4	1.1	0.3	-1.5
Saturday	0.3	-0.2	0.6	0.0

A differenza di un One Hot Encoding, in cui un giorno della settimana può essere solo un singolo valore, gli Embeddings trasformano il giorno della settimana in un concetto a  $k$  dimensioni (nella fattispecie, 4). Dopo aver addestrato il modello, si può scoprire che questa tabella contiene un significato semantico. Ad esempio, sabato e domenica potrebbero essere più strettamente correlati di sabato e mercoledì.

L'idea è quella di creare degli Embeddings per ciascuna variabile categoriale a disposizione, concatenarli e allenare una rete neurale profonda per apprendere dagli Embeddings costruiti.

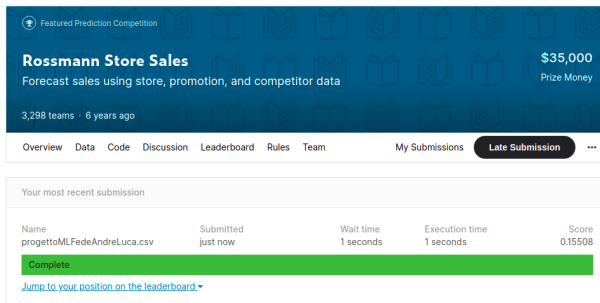
In questa sezione verranno esclusi, per semplicità, le variabili continue e si allenerà la rete solo sulle variabili esplicative categoriali.

È stata utilizzata la libreria **Keras** per creare gli Embeddings, utilizzando lo specifico layer già implementato nel pacchetto, e per allenare una rete neurale con 3 livelli nascosti (rispettivamente con un numero di neuroni pari a: 128, 64, 32).

Purtroppo i risultati sono poco soddisfacenti, per questo si è deciso di scartare questo metodo.

## 8 CONCLUSIONI E POSSIBILI SVILUPPI

Nonostante le limitate capacità computazionali che si avevano a disposizione per la risoluzione del topic in questione, è stato raggiunto un risultato, sul test set, pari a **0.155808** (ottenuto tramite l'utilizzo di **SuperLearner (GLM) Ensemble**), con una distanza dalle prime posizioni della **Leaderboard** di Kaggle di 5-6 punti percentuali, senza però che si utilizzassero dataset aggiuntivi, messi a disposizione a competizione iniziata, riguardanti: la regione, la città, le temperature e il meteo registrati per ciascuno store nella specifica data.



Featured Prediction Competition				
Rossmann Store Sales				\$35,000
Forecast sales using store, promotion, and competitor data				Prize Money
3,298 teams · 6 years ago				
Overview	Data	Code	Discussion	Leaderboard
Rules	Team	My Submissions	Make Submission	...
Your most recent submission				
Name	Submitted	Wait time	Execution time	Score
progettoMLFedeAndreLuca.csv	just now	1 seconds	1 seconds	0.155808
Complete				
<a href="#">Jump to your position on the leaderboard</a>				

I metodi di Machine Learning per l'analisi delle serie storiche si sono rivelati efficaci nell'interpretare e prevedere dati di natura temporale. Datasets come quello in esame, caratterizzati da un'elevata dimensionalità e da un cospicuo numero di variabili categoriali con alta cardinalità sarebbero stati praticamente impossibili da trattare utilizzando i tradizionali metodi ARIMA.

I risultati ottenuti con i *Tree-based method* sono quelli più soddisfacenti; essi sono stati scelti proprio perchè nelle competizioni Kaggle sono stati gli algoritmi *shallow* che hanno performato meglio. Inoltre, sarebbe stato interessante approfondire i modelli basati su Reti Neurali e sfruttare quelle che sono le architetture più recenti per lo studio di dati temporali: le reti neurali ricorrenti (LSTM, GRU, ...), proprio nell'ottica di utilizzare il dataset come benchmark per modelli e tecniche diverse.

## REFERENCES

[1] Aiello, S., Eckstrand, E., Fu, A., Landry, M. & Aboyoun, P. Machine learning with r and h2o. *H2O booklet* **550** (2016).

[2] Bischl, B. *et al.* mlr: Machine learning in r. *The J. Mach. Learn. Res.* **17**, 5938–5942 (2016).

[3] Massicotte, P., Eddelbuettel, D. & Massicotte, M. P. Package 'gtrends'. (2016).

[4] Köppen, M. The curse of dimensionality. In *5th On-line World Conference on Soft Computing in Industrial Applications (WSC5)*, vol. 1, 4–8 (2000).

[5] Pargent, F., Pfisterer, F., Thomas, J. & Bischl, B. Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. *arXiv preprint arXiv:2104.00629* (2021).

[6] Pavlyshenko, B. Using stacking approaches for machine learning models. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, 255–258 (IEEE, 2018).