ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

Dipartimento di Informatica – Scienza e Ingegneria DISI

Laurea Magistrale in Ingegneria Informatica

TESI DI LAUREA

in

Linguaggi e Modelli Computazionali

Tecniche di Machine Learning nei Modelli previsionali per la pianificazione logistica

Candidato:

Luca Burroni

Relatore:

Prof. Enrico Denti

Correlatori:

Dott. Luca Mastellari

Dott. Fabio Gravina

Anno Accademico 2022/2023

Sessione III

Sommario

Introduzione

Negli ultimi tempi stiamo assistendo a una sempre più rapida evoluzione nel campo dell'intelligenza artificiale, in particolar modo il Machine Learning ha assunto un ruolo di rilievo, guadagnando crescente attenzione per la sua capacità di apprendere da dati passati, identificare modelli e anticipare comportamenti futuri.

L'adozione di queste tecniche si è diffusa in molteplici settori industriali, rivoluzionando i processi decisionali e consentendo una previsione più accurata e tempestiva.

La logistica, motore vitale dell'efficienza operativa e della catena di approvvigionamento, non poteva non essere coinvolta in questa trasformazione. Attraverso l'implementazione di modelli predittivi basati su Machine Learning, le aziende possono ottimizzare la pianificazione logistica, migliorare la gestione delle risorse e ridurre i costi operativi.

Obiettivo di questa tesi è quindi indagare come l'applicazione di tecniche di Machine Learning nei modelli previsionali per la pianificazione logistica, a partire da numerosi dati storici, possa migliorare la stima dei tempi di evasione delle attività di carico e scarico presso magazzini logistici.

In tal modo si mira a fornire agli operatori logistici, alle aziende e agli studiosi un quadro completo delle opportunità e delle sfide associate all'implementazione di queste tecniche avanzate, aprendo la strada a future innovazioni e miglioramenti nel settore.

1. Dominio del problema

I magazzini logistici sono organizzati in *baie*, aree specifiche destinate alle *operazioni* di carico e scarico delle merci. Il numero delle baie varia da magazzino a magazzino, ma ovviamente è sempre un numero limitato. Per questa ragione, unita all'alto numero di operazioni da effettuare giornalmente, l'accesso dei mezzi alle baie è un problema di gestione delle risorse che è fondamentale ottimizzare.

La stima del tempo di evasione delle operazioni da effettuare diventa quindi un dato cruciale per una corretta pianificazione. Ovviamente ogni operazione necessita di un diverso tempo di evasione, il quale dipende da numerosi fattori.

Il lavoro svolto in questa tesi si inserisce come parte integrante di un software già esistente, Load Manager [1], una web application, sviluppata dall'azienda OltreSolutions, che permette di pianificare e gestire gli avanzamenti delle attività di carico e scarico presso le piattaforme logistiche.

Esso costituisce, una volta configurato, un "digital twin" che mira a catturare tutte le regole e le specificità del processo gestito da un certo magazzino, fornendo un servizio di portineria virtuale 24/7 attraverso il quale è possibile raccogliere prenotazioni dialogando direttamente con i partner (clienti/fornitori/trasportatori).

La portineria virtuale di Load Manager si fa carico di popolare la pianificazione operativa delle attività di carico/scarico, trovando il miglior compromesso tra le necessità dei partner e le specificità del business [2].

I parametri considerati per la stima sono quelli forniti al momento della prenotazione: a seguire Load Manager stima il tempo di evasione.

1.1. Parametri richiesti per effettuare una prenotazione

Per effettuare una prenotazione mediante Load Manager occorre passare attraverso quattro step. Le informazioni utili per la stima del tempo di evasione dell'operazione sono raccolte durante i primi tre.

Durante il primo, come mostrato in Figura 1, viene richiesto di selezionare in quale magazzino si vuole effettuare l'operazione.

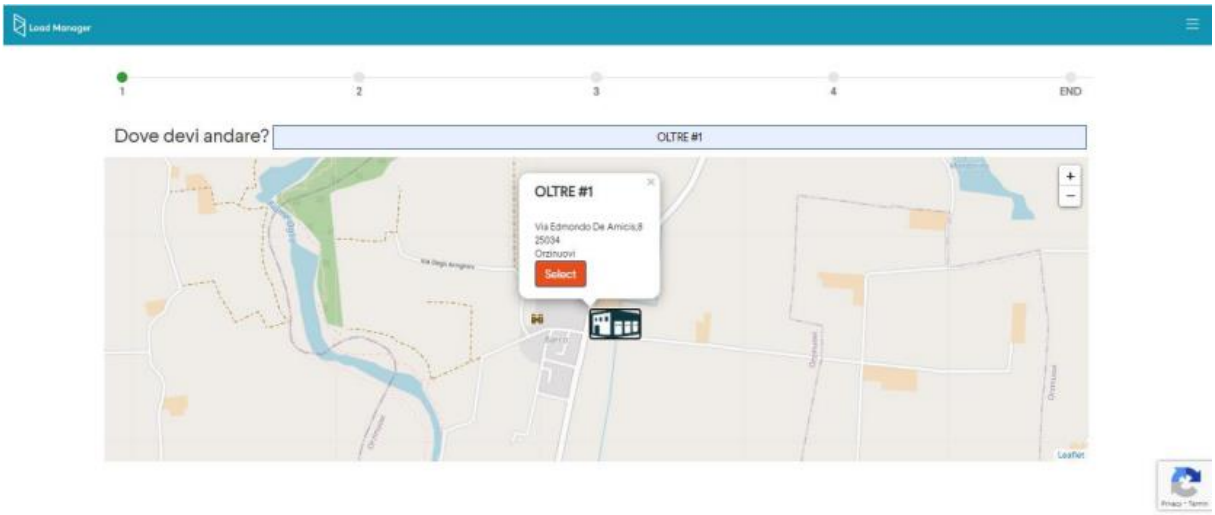

Figura 1 - Step 1 Prenotazione

Nel secondo invece, come si può vedere in Figura 2 e Figura 3, vengono richieste diverse informazioni, nello specifico:

- Data e ora di arrivo previsto
- Tipo di mezzo (selezione da una lista personalizzata per ogni magazzino)
- Email utente
- Una o più attività, per le quali occorre specificare:
    - *Azione*: carico o scarico
    - Tipologia di *handling unit* / *item* (selezione da una lista personalizzata per ogni magazzino)
    - Quantità di handling unit di tale tipologia
    - Modalità di spedizione (selezione da una lista personalizzata per ogni magazzino)

- Provenienza o destinazione (selezione da una lista personalizzata per ogni magazzino)

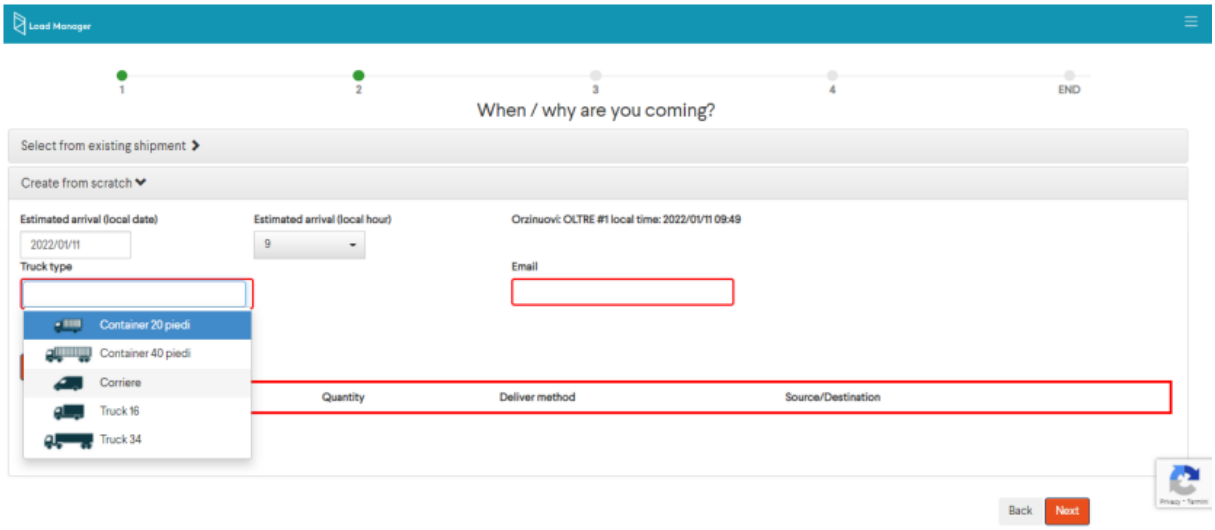

Figura 2 - Step 2 Prenotazione: TruckType

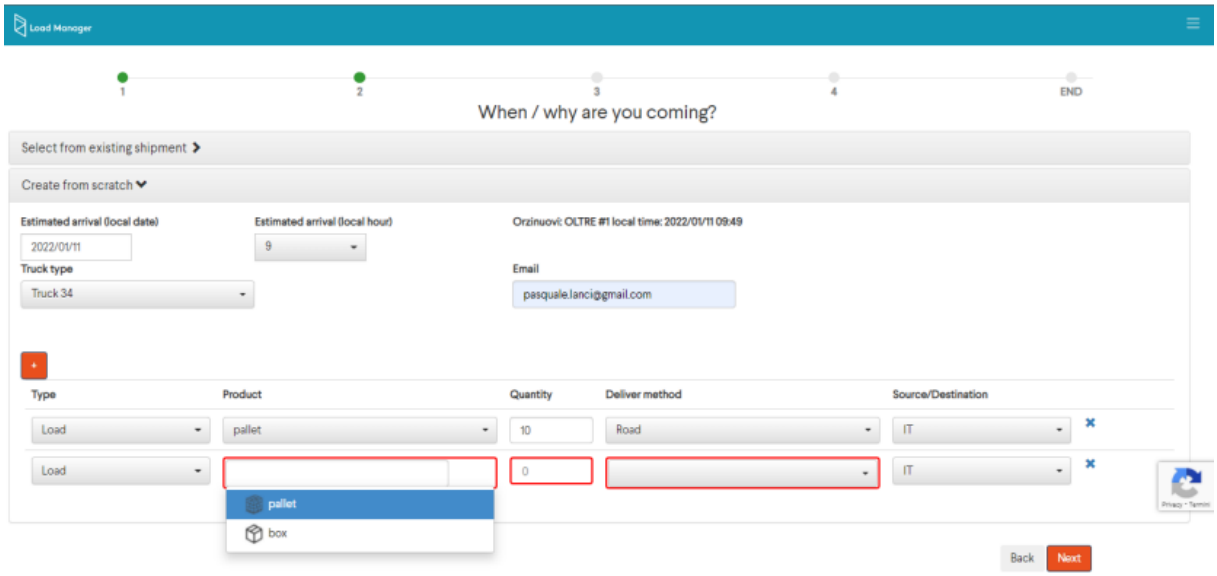

Figura 3 - Step 2 Prenotazione: Attività

La possibilità di inserire più attività per ogni operazione deriva dal fatto che un mezzo potrebbe dover sia scaricare che caricare, oppure dover caricare (o scaricare) più tipologie di handling unit (item) differenti, ad esempio 20 pallet e 10 box.

Il terzo step (Figura 4) è dedicato a eventuali informazioni aggiuntive, il cui insieme è definito in modo personalizzato sulla base delle esigenze di ogni singolo magazzino.

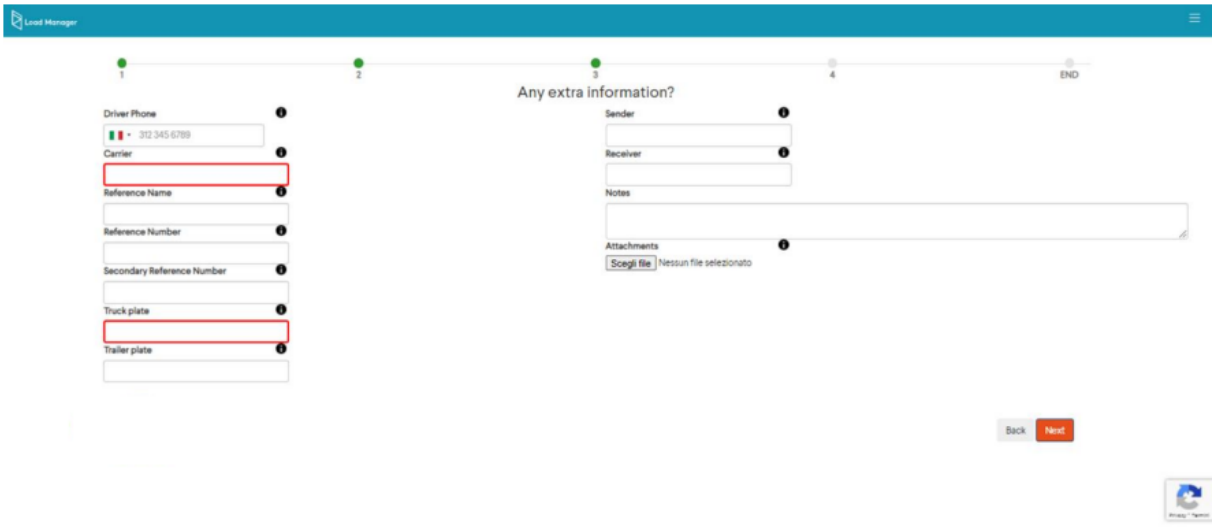

Figura 4 - Step 3 Prenotazione

1.2. Stima dei tempi e scelta dello slot

Una volta raccolte le informazioni richieste nei primi tre step, il tempo di evasione viene stimato immediatamente e utilizzato per il quarto e ultimo step, illustrato in Figura 5.

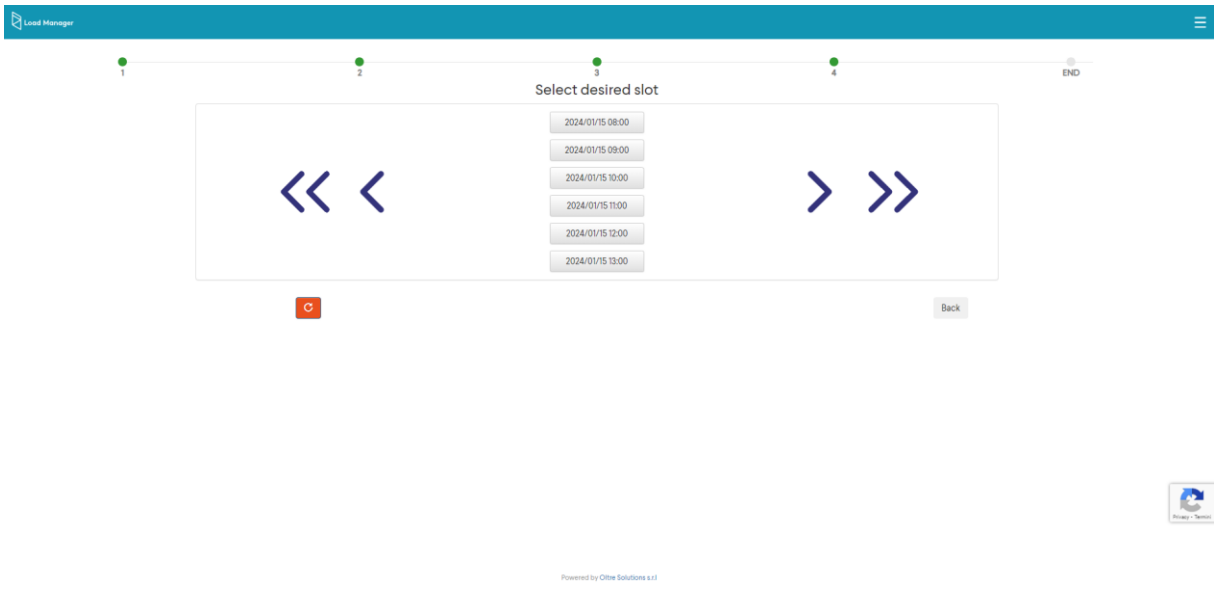

Figura 5 - Step 4 Prenotazione

Questo step prevede di selezionare uno slot temporale tra quelli proposti a partire dalla data/ora di arrivo previsto precedentemente indicata (step 2). Tali slot temporali hanno orario e durata fissi, scelti dal magazzino.

Il sistema controlla altresì che dalla data/ora dello slot il tempo previsto di evasione sia collocabile nella baia e non ci siano altre prenotazioni/pause/chiusure collidenti. Dunque, nel caso in cui la durata prevista di un'operazione superi la durata di uno slot, saranno selezionati (e proposti) soltanto gli slot liberi in cui anche il successivo (o i successivi) risulti libero.

Una volta selezionato lo slot tra quelli proposti, il sistema notifica la corretta creazione della prenotazione e la aggiunge nel relativo slot (e baia) con la relativa durata, come si può vedere dal Visual Planner mostrato in Figura 6.

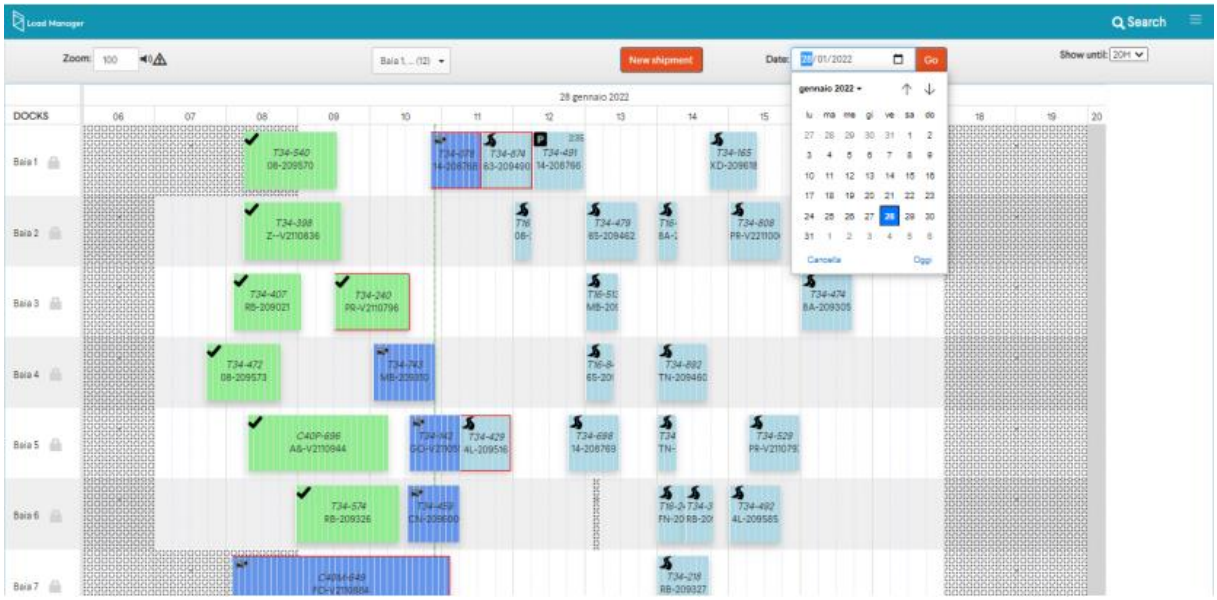

Figura 6 - Visual Planner: ogni rettangolo rappresenta una prenotazione con la relativa durata stimata

1.3. Stima attuale

Il lavoro svolto in questa tesi ha come obiettivo quello di indagare se, attraverso l'uso di modelli di *Machine Learning*, sia possibile migliorare la stima dei tempi di evasione delle operazioni rispetto alla stima attualmente effettuata da Load Manager, basata sulla formula generalizzata seguente:

$$d_s = \sum_{i=0}^{n} t_i * tot_i$$

Dove:

- d_s= tempo previsto di evasione dell'operazione
- n = numero di tipologie *item-azione* (item o handling unit) che compongono l'operazione. Con item-azione si intende una determinata azione da svolgere per una determinata tipologia di item, ad esempio carico di box o scarico di pallet. Occorre notare che le azioni di carico e scarico di una stessa determinata tipologia di item sono considerate tipologie item-azione differenti (solitamente hanno tempi differenti)
- t_i = tempo previsto (aggiornato periodicamente) per eseguire una determinata azione su una singola unità di una determinata tipologia di item in un determinato magazzino
- tot_i = numero di item di una determinata tipologia per i quali eseguire una determinata azione

Il tempo t_i, per ogni tipologia item-azione, viene stabilito dal magazzino in fase di configurazione del sistema; dopodiché, contestualmente all'aumento di operazione evase, viene periodicamente aggiornato, così da migliorare la stima, mediante la seguente procedura:

- Si considerano tutte le operazioni già evase che contengono solo la tipologia item-azione considerata (e.g. solo scarico di soli pallet) nel magazzino considerato
- Si calcola, per tali operazioni, il tempo medio impiegato per singolo item come rapporto tra la somma delle durate complessive e il numero complessivo di item: $t_{i_{new}} = \frac{\sum_{i=1}^{n} d_{eff_i}}{\sum_{i=1}^{n} n_{item_i}}$ dove, per una singola operazione, d_{eff} rappresenta la durata effettiva e n_{item} il numero di item

- Si calcola il nuovo t_i come $t_i = t_{i_{old}} + [(t_{i_{new}} - t_{i_{old}}) * (\frac{m}{100})]$ dove m è il valore correttivo (fra 0 e 100) che determina il peso della correzione.

Occorre ribadire che, nelle formule soprastanti, t_i non identifica il tempo previsto per un certo item in assoluto, ma relativamente all'azione da svolgere (item-azione).

2. Data Preparation

I numerosi dati storici, relativi alle operazioni concluse, sono preliminarmente organizzati in appositi file di tipo “comma separated (.csv)”, suddivisi per magazzino: ciò in quanto, per una scelta progettuale, si è deciso di creare un diverso modello di Machine Learning per ogni magazzino.

Questa scelta deriva da due fattori:

- Magazzini diversi potrebbero lavorare con tipologie di item differenti
- Una stessa tipologia di item potrebbe rappresentare qualcosa di diverso tra un magazzino e un altro.

In queste condizioni utilizzare un unico modello (con un’eventuale *feature [3]* a discriminare il magazzino) impatterebbe negativamente sull’efficienza del modello, poiché:

- Aumenterebbe il numero di feature: una per ogni tipologia item-azione (contenente il numero di item di una determinata tipologia per i quali eseguire una determinata azione) selezionabile in almeno un magazzino
- Nel caso di tipologie item-azione selezionabili in pochi magazzini, o in magazzini con poche operazioni evase rispetto al totale, si avrebbero le relative colonne con valore quasi sempre zero, complicando l’attribuzione del giusto peso a tali feature da parte del modello
- L’attribuzione del giusto peso a tali feature da parte del modello sarebbe complicata anche dal possibile diverso significato che differenti magazzini potrebbero attribuire a una stessa tipologia di item.

La Tabella 1 illustra questi aspetti: essa riporta i tempi medi per singolo item di una certa tipologia item-azione nei vari magazzini considerati, calcolati come rapporto tra la durata effettiva di un’operazione evasa e il numero di item caricati o scaricati in tale operazione. Chiaramente, anche qui, sono considerate solo le operazioni che prevedono una sola tipologia item-azione (colonna MonoItem): le righe della tabella in cui il valore della colonna MonoItem è basso (poche operazioni considerate) hanno un basso valore statistico.

Si può osservare come magazzini diversi possano impiegare tempi molto diversi per evadere un elemento della stessa tipologia item-azione.

In questa tabella l'unica operazione di *Data Cleaning* effettuata è la rimozione degli *outlier* [8] considerati estremi, ovvero le operazioni con durata effettiva minore di due minuti o maggiore di 300, in quanto palesemente *sporchi* per errore umano.

È possibile osservare anche come, nella maggior parte dei magazzini, le operazioni consistano nello svolgere principalmente una determinata azione per una determinata tipologia di item (e.g. nel magazzino DSE_BG03 si ha quasi esclusivamente scarico di pallet). Questa è una caratteristica piuttosto comune dei magazzini logistici.

Magazzino	Item	Count	MonoItem	Mean t_i (min)
TCN_FC01	PalletUnload	23492	22591	1.81
	BoxUnload	6208	5381	1.81
	FTLUnload	709	633	27.29
TCN_FC02	FTLLoad	8184	8184	54.28
	LTLLoad	713	713	34.89
DSE_BG03	PalletLoad	8664	8664	1.79
	PalletUnload	119	119	1.54
	EmptyPalletLoad	1	1	2.18
DSE_BO01	PalletLoad	12067	12066	2.89
	PalletUnload	4	3	3.97
	EmptyPalletLoad	4	3	4.02
DSE_MB01	PalletLoad	7768	7758	2.23
	PalletUnload	497	489	1.56
	FTLLoad	2	2	0.5
	EmptyPalletLoad	1	1	0.13
	EmptyPalletUnload	4	1	0.09
DVS_BO01	PalletLoad	1128	473	1.62
	PalletUnload	6988	5744	1.32
	EmptyPalletLoad	597	9	1.35
	EmptyPalletUnload	30	23	1.2
FLR_FL01	PalletLoad	2606	443	5.28
	PalletUnload	36	4	29.62
	CampioneLoad	1348	75	20.02
	CassaLoad	2047	71	15.56
	CavallettoLoad	76	4	18.84
	EmptyCassaUnload	40	4	13.05

Tabella 1 - Tempo medio per singolo item di ogni tipologia Item-Azione in ogni magazzino (filtro outlier solo estremi)

2.1. Feature Selection

Come introdotto all'inizio del Capitolo 2, i dati storici, relativi alle operazioni concluse, sono organizzati in appositi file "comma separated (.csv)" suddivisi per magazzino.

Per preparare il *Dataset* [3], da usare come input per i modelli di Machine Learning, occorre sia identificare la colonna *target* [4], che selezionare le colonne da utilizzare come *feature* [4].

La colonna target è chiaramente quella che contiene la durata effettiva di una prenotazione evasa, poiché il modello di Machine Learning dovrà apprendere come stimare tale durata a priori.

Per quanto riguarda la selezione delle feature occorre prendere in considerazione diversi fattori:

- Le sole colonne da considerare sono quelle contenenti informazioni fornite in fase di prenotazione, in quanto tali informazioni sono le sole certamente disponibili al momento della stima: vanno invece escluse le informazioni note solo a posteriori, quali ad esempio il tempo di attesa effettivo di un mezzo prima dell'accesso alla baia assegnata
- Le colonne aventi molti dati mancanti, ovvero quelle relative a informazioni da fornire facoltativamente, non vengono selezionate
- Per quanto riguarda le colonne contenenti dati categorici, per la selezione si adotta una metodologia basata sui due fattori seguenti:
    - Analisi dei *boxplot*: i boxplot [5] sono grafici statistici che mostrano la distribuzione dei valori della variabile dipendente (continua), in questo caso la durata effettiva, nelle categorie della variabile indipendente (discreta). Ogni boxplot fornisce una chiara rappresentazione grafica della variabilità e della mediana nelle varie categorie. Se la distribuzione dei valori del target differisce tra le varie categorie di una certa colonna, quella colonna è discriminante e, quindi, una possibile valida feature
    - Test ANOVA: l'analisi della varianza (ANOVA) [6] è una procedura statistica che permette di valutare la significatività delle differenze medie nei valori del target tra le diverse categorie. Un valore elevato dell'*F-statistic*, accompagnato da un basso valore *p*, indica la presenza di differenze statisticamente significative nei valori del target tra almeno alcune categorie. Tali condizioni suggeriscono che la colonna in analisi è discriminante e, quindi, una possibile valida feature.

- Le uniche colonne contenti dati numerici non categorici sono le colonne relative alle varie tipologie item-azione (e.g. scarico di pallet), contenenti il numero di item, di quella determinata tipologia, per i quali è stata effettuata quella determinata azione. Tali colonne vengono tutte selezionate come feature, indipendentemente da possibili valori statistici considerabili, come ad esempio l'indice di correlazione di Pearson [7] con il target, per ragioni di scalabilità: come spiegato all'inizio del Capitolo 2 e osservabile guardando la Tabella 1, nei magazzini logistici solitamente si svolge principalmente una determinata azione per un determinato item. Questo comporta che le colonne relative alle varie tipologie item-azione poco frequenti, ma selezionabili in fase di prenotazione, in un dato magazzino hanno spesso valore zero e, quindi, un basso indice di correlazione di Pearson con il target. Se non selezionassimo le colonne relative a tali tipologie item-azione, dovremmo cambiare il modello di Machine Learning nel caso in cui il dato magazzino cambiasse la propria tendenza.

Fatta eccezione per le colonne relative alle tipologie item-azione, che possono differire tra i vari magazzini, le feature selezionate sono le medesime per tutti i magazzini considerati.

Di seguito è riportato l'elenco di tutte le colonne (feature selezionate + target) che compongono il Dataset da usare come input per i modelli di Machine Learning:

- **Hour**: ora di arrivo al magazzino tra quelle disponibili
- **DayOfWeek**: giorno della settimana (lun-ven)
- **Month**: mese, codificato come coppia di mesi per aumentare la differenza nei valori del target tra le varie categorie
- **TruckType**: tipo di mezzo
- **Action**: carico, scarico o misto
- **MultiItem**: specifica se l'operazione include una sola tipologia di item o più
- **Sender**: mittente. Si mantengono i cinque con il maggior numero di occorrenze e si codificano gli altri come 'Others', in modo che le categorie abbiano un reale rilevanza statistica
- **Receiver**: ricevente. Si usa lo stesso procedimento usato per la feature Sender
- **Carrier**: trasportatore. Si usa lo stesso procedimento usato per le feature Sender e Receiver
- *Feature item-azione* (e.g. **PalletLoad**, **PalletUnload**, **BoxLoad** e **BoxUnload**): come già ampiamente discusso, per ogni tipologia item-azione si ha una feature che identifica

per quanti item di una determinata tipologia si esegue una determinata azione all'interno dell'operazione

- **EffDuration**: Durata effettiva dell'operazione (nota solo a posteriori). Colonna target, valore da imparare a stimare.

Per facilitare la comprensione si mostrano di seguito i boxplot relativi ad alcune feature categoriche per uno dei magazzini considerati (DSE_BG03). In particolare: TruckType (Figura 7), Hour (Figura 8), Month (Figura 9) e Carrier (Figura 10). Per ognuno dei grafici è possibile osservare come la distribuzione dei valori della variabile dipendente (target) differisca tra le varie categorie della feature in analisi.

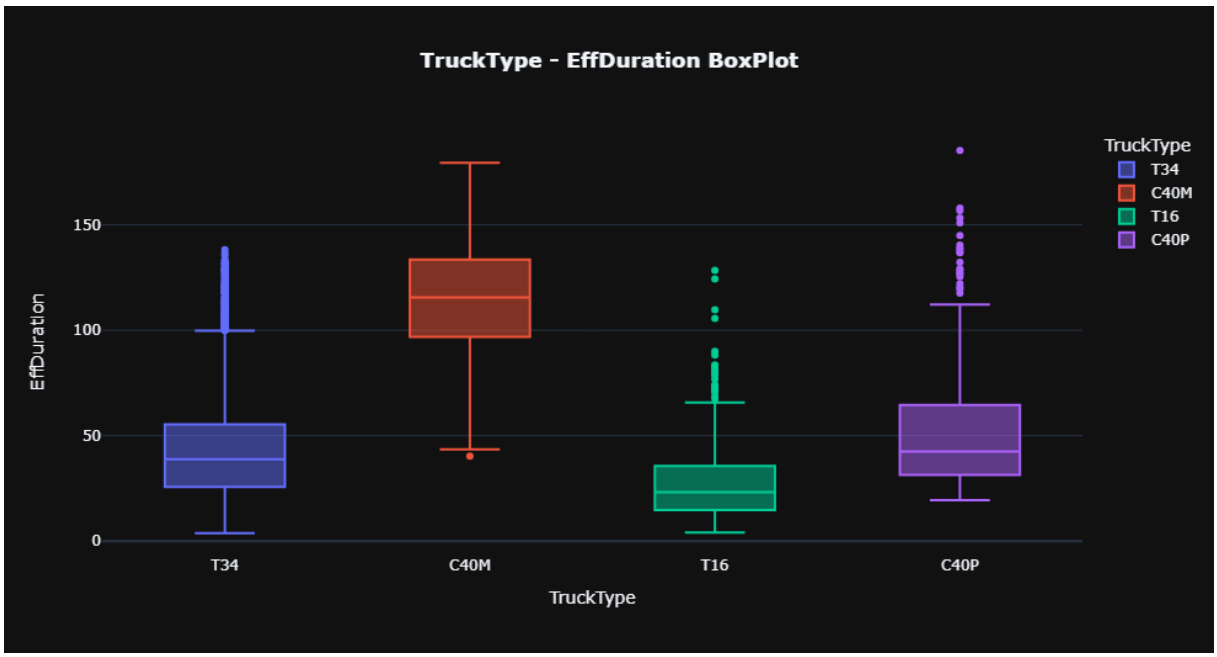

Figura 7 - BoxPlot TruckType - EffDuration: mostra come si distribuiscono i valori di EffDuration rispetto alle varie categorie di TruckType

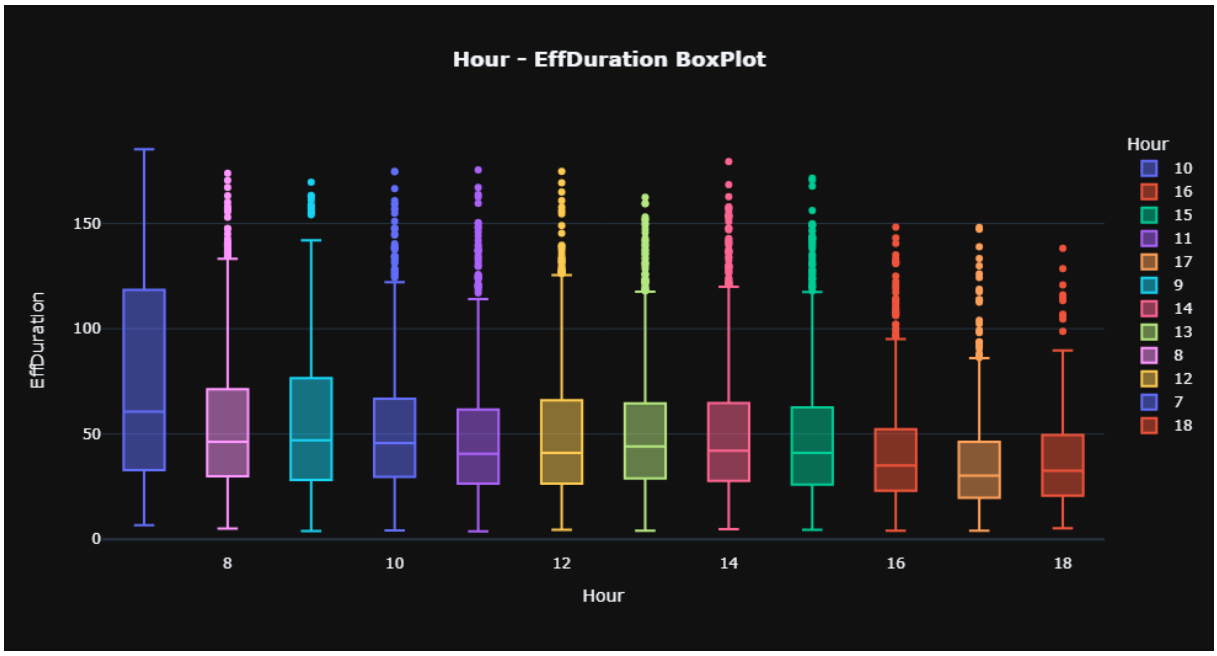

Figura 8 - BoxPlot Hour-EffDuration

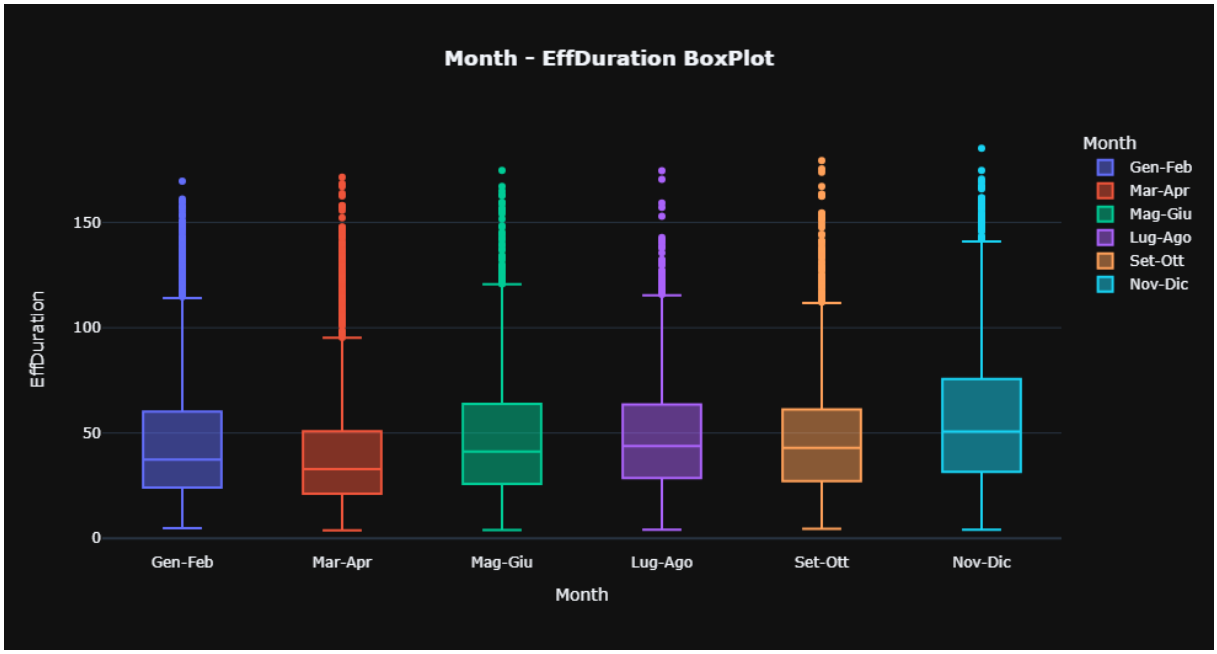

Figura 9 - BoxPlot Month – EffDuration

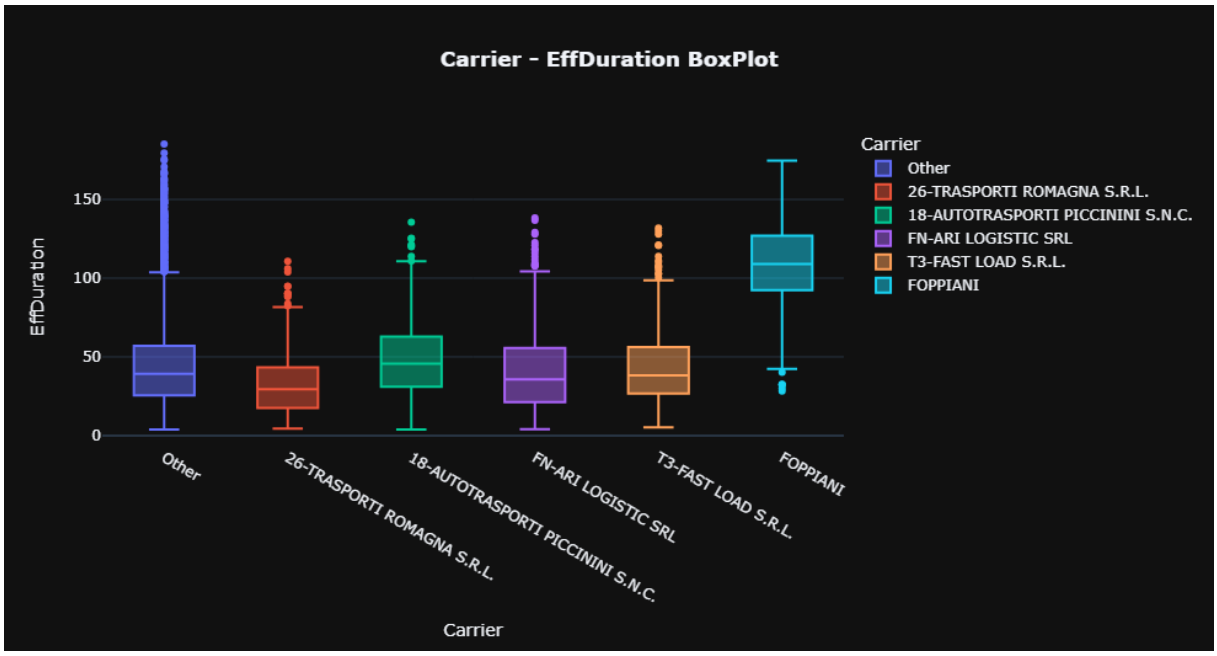

Figura 10 - BoxPlot Carrier – EffDuration

2.2. Data Cleaning

Il problema principale dell'utilizzo di modelli di Machine Learning in questo dominio risiede nel fatto che i dati risultano essere molto sporchi, in quanto forniti manualmente da operatori logistici e, di conseguenza, molto esposti a possibili errori umani (potenzialmente diverso sarebbe se, in altro scenario o in futuro, fossero rilevati tramite sensori o altra tecnologia).

Inoltre, il dato più soggetto a errori è proprio la durata effettiva di un'operazione, ovvero il valore target. Infatti, mentre i dati utilizzati come feature sono inseriti in fase di prenotazione (molti di questi selezionati da una lista) e, quindi, in un contesto tipicamente tranquillo, la durata effettiva (target) è calcolata come la differenza tra il momento di inizio operazione e il momento di fine operazione inseriti da un operatore logistico sul momento, in un contesto inevitabilmente caotico.

Si rende dunque inevitabile una fase di pulizia del Dataset, ottenuto come spiegato nel Sottocapitolo 2.1, per poterlo utilizzare come input per i modelli di Machine Learning.

2.2.1. Filtraggio degli outlier estremi

Come prima cosa conviene quindi effettuare un filtraggio sulla base della conoscenza del dominio: vengono stabiliti dei range di valori consentiti per alcune delle colonne (feature + target) e eliminate dal Dataset le righe (operazioni evase) con valori di tali feature fuori dal relativo range, considerati outlier [8]. Ad esempio si eliminano le righe con durata effettiva inferiore a due minuti o superiore a cinque ore, ritenute assolutamente improbabili/impossibili nella realtà: in Figura 11 e Figura 12 sono riportati gli *istogrammi* [9] che mostrano la distribuzione dei valori della durata effettiva, in uno dei magazzini considerati (DSE_BO01), prima e dopo questo tipo di filtraggio. Come si può notare, nel secondo caso la scala dell'asse orizzontale è limitata a 300, e il picco precedentemente presente vicino allo 0 viene eliminato.

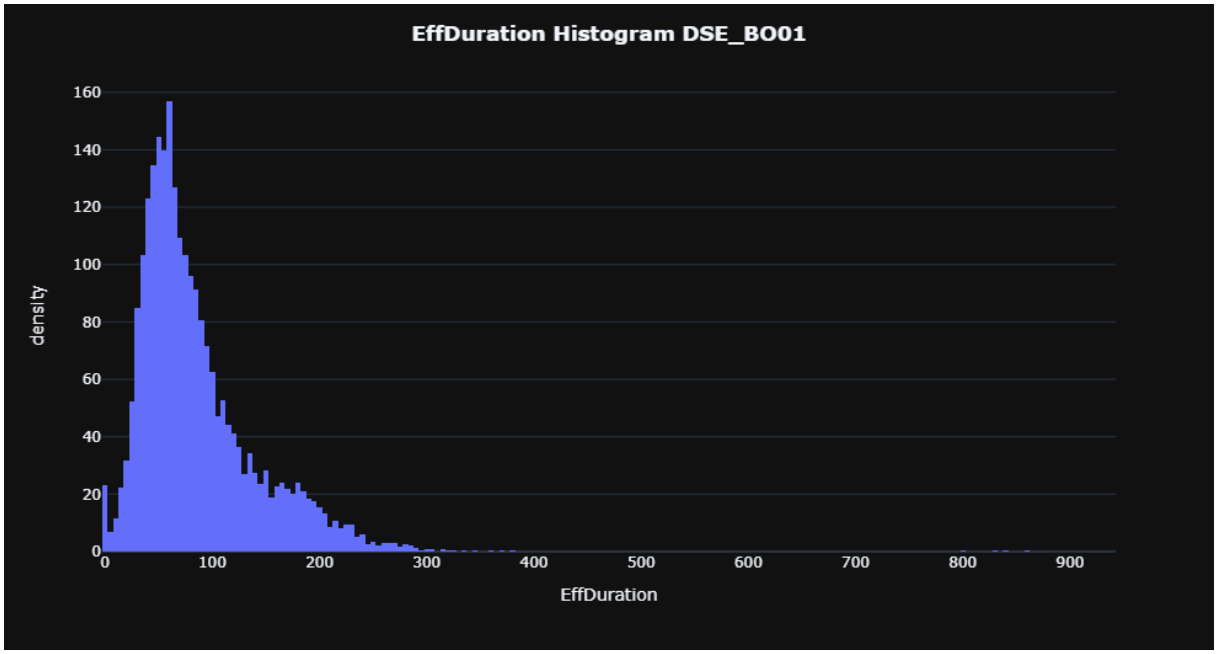

Figura 11 - Istogramma EffDuration: mostra la distribuzione dei valori della durata effettiva per il magazzino DSE_BO01

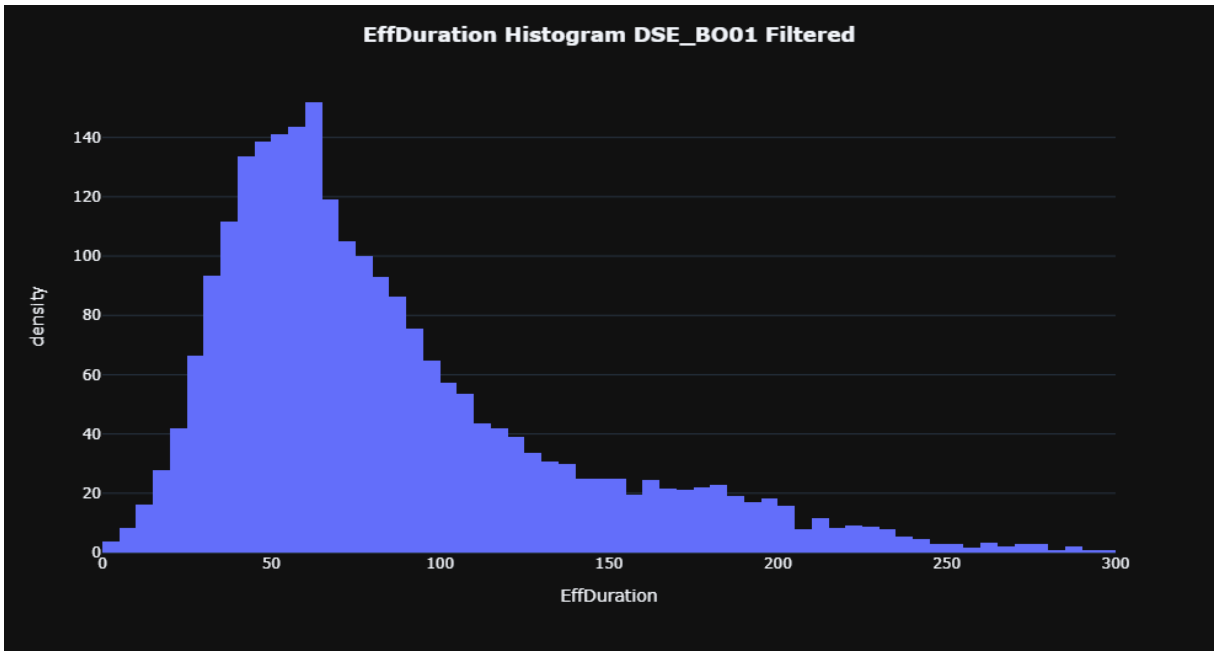

Figura 12 - Instogramma EffDuration dopo aver eliminato i valori fuori dal range

2.2.2. Filtraggio avanzato

Sebbene le tecniche spiegate nel Sottocapitolo 2.2.1 permettano di avere un Dataset più pulito, è opportuno, in questo dominio, effettuare una pulizia dei dati più avanzata.

Fin qui, infatti, l'identificazione (e successiva rimozione) degli outlier è stata limitata a contesti monodimensionali, guardando solo la distribuzione dei valori all'interno delle varie colonne. Tuttavia è possibile filtrare i dati operando in contesti multidimensionali.

Dalla conoscenza del dominio, unita a osservazioni statistiche, risulta che le feature più discriminanti per il valore della durata effettiva di una prenotazione sono la tipologia di mezzo (TruckType) e le feature item-azione.

Essendo TruckType una feature categorica, è possibile ottenere, per ogni categoria, uno *scatterplot* [10], grafico che visualizza la relazione tra due variabili mediante l'uso di punti disposti su un piano cartesiano, avente sull'asse orizzontale i valori di una determinata feature item-azione e sull'asse verticale i valori della colonna target (durata effettiva). In Figura 13 e in Figura 14 sono riportati tali scatterplot per la feature item-azione PalletLoad (numero di pallet caricati), con TruckType C40M (Figura 13) e T34 (Figura 14), in uno dei magazzini considerati (DSE_BG03).

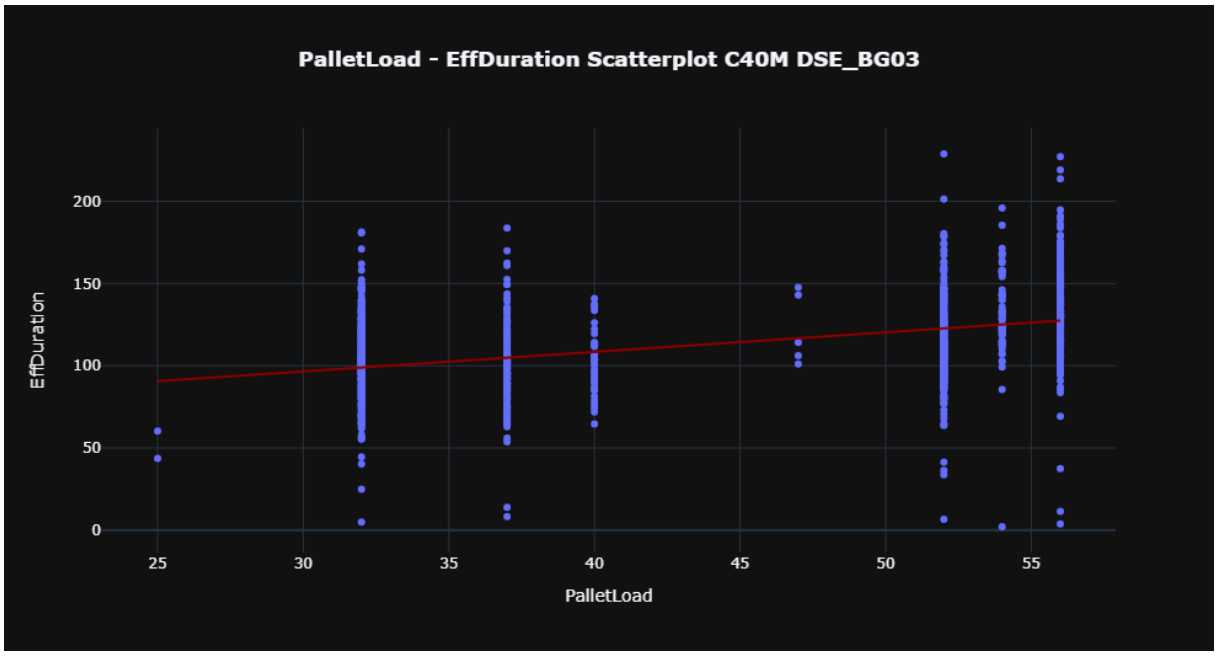

Figura 13 - Scatterplot PalletLoad - EffDuration con TruckType C40M nel magazzino DSE_BG03. Retta di regressione calcolata usando la tecnica OLS

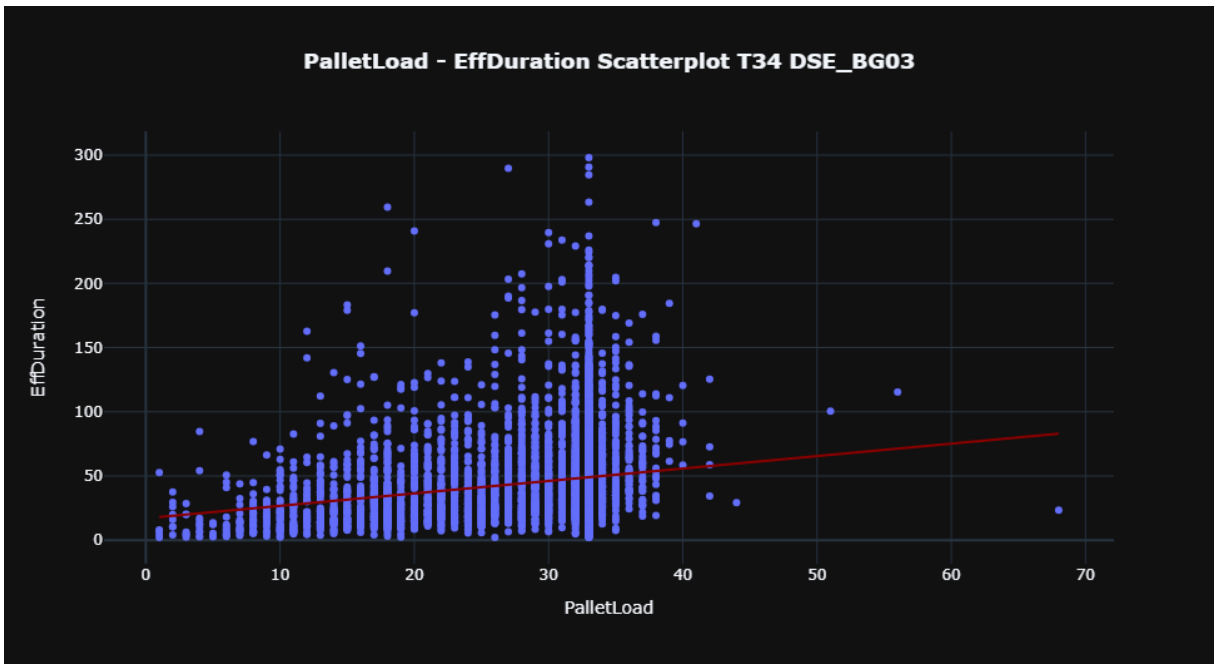

Figura 14 - Scatterplot PalletLoad - EffDuration con TruckType T34 nel magazzino DSE_BG03.

Occorre specificare che per il momento vengono considerate solo le operazioni *mono-item* (una sola tipologia item-azione). Questo in quanto la relazione tra il valore di una feature item-azione e la durata effettiva dell'intera operazione è significativa solo in questo caso. Ad esempio, se

un’operazione ha durata effettiva di 30 minuti e consiste nel solo scarico di 30 pallet (mono-item), la relazione tra il valore della feature item-azione pallet-scarico, che vale 30, e la durata effettiva (pure 30), è significativa. In tal caso l’analisi degli scatterplot (uno per TruckType), che mostrano come varia il valore della durata effettiva al variare del valore della feature pallet-scarico, ha significato. Se, invece, l’operazione avesse natura *multi-item*, ad esempio consistesse nello scaricare 20 pallet e caricarne 30, la sola relazione tra il valore della feature pallet-scarico, che varrebbe 20, e la durata effettiva (ipotizzata sempre pari a 30) non sarebbe significativa, poiché quest’ultima includerebbe anche la parte relativa alla feature pallet-carico.

Per evidenziare meglio la relazione fra due variabili un tipico modo è quello di calcolare la retta di regressione¸ intesa come quella retta che cerca di modellare la relazione tra le due variabili. Una volta definita, essa può essere utilizzata per predire il valore della variabile dipendente in funzione della variabile indipendente. Nel nostro caso (Figura 13 e Figura 14), emerge che non poche operazioni evase risultano avere una durata effettiva piuttosto lontana dalla retta di regressione ottenuta mediante metodo dei minimi quadrati (OLS) [11].

In questo contesto, è possibile affermare che più un valore è lontano dalla retta di regressione, più probabilmente è determinato da un errore umano e, quindi, da scartare in quanto outlier.

Al fine di automatizzare tale operazione di identificazione degli outlier in contesti multidimensionali, possono essere prese in considerazione misure statistiche applicabili al caso multidimensionale, in particolare la distanza di Mahalanobis [12] [13] e la distanza di Cook [14] [15]. La prima è una tecnica statistica che può essere usata per misurare la distanza di un punto dal centro di una distribuzione normale multivariata [13] [16], mentre la seconda è una funzione comunemente usata per stimare l'influenza di un singolo punto in un'analisi di regressione ai minimi quadrati [14].

Tuttavia, in entrambi i casi, il risultato non si rivela del tutto soddisfacente, per i motivi seguenti:

- Nel caso della distanza di Mahalanobis, sebbene la durata effettiva abbia quasi sempre una distribuzione riconducibile a una curva normale mediante trasformazioni, non si può dire lo stesso delle feature item-azione, come si può vedere chiaramente, ad esempio, in Figura 13. Per questa ragione, l’ellissi generata attraverso la distanza di Mahalanobis dal centro, finisce quasi sempre per escludere dei valori che sono in realtà corretti. Ciò è mostrato in Figura 15, dove è riportato lo stesso scatterplot di Figura 14, nel caso in cui si utilizzi la distanza di Mahalanobis per individuare gli outlier. Qui, ad

esempio, vengono considerati outlier in modo incondizionato tutti i valori vicini allo zero

- Nel caso della distanza di Cook è possibile considerare outlier i punti con una distanza di Cook superiore a una certa soglia. Tuttavia, in questo contesto, questa procedura ha poco significato matematico, in quanto parte da un'analisi di regressione ai minimi quadrati molto condizionata dai numerosi outlier presenti. In Figura 16 è riportato lo stesso scatterplot di Figura 14, nel caso in cui si utilizzi la distanza di Cook per individuare gli outlier. Come si può notare, molti valori che nei fatti sono considerabili inverosimili in quanto piuttosto lontani dalla retta di regressione (Figura 14) non vengono identificati come outlier.

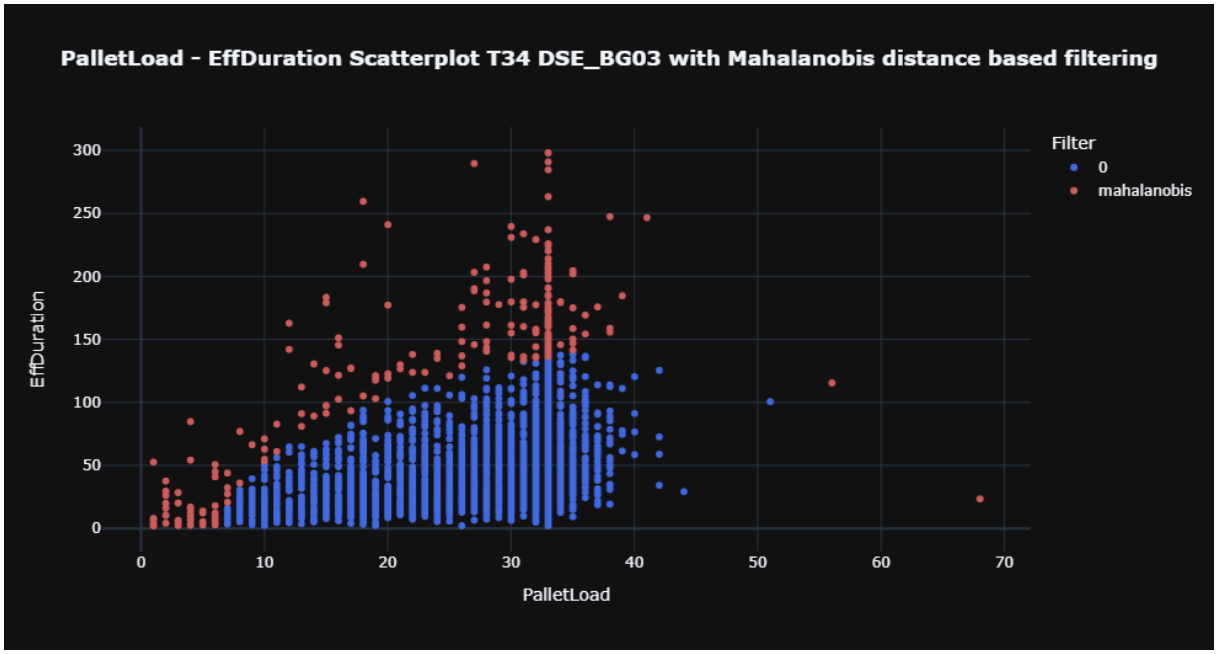

Figura 15 - Filtraggio basato sulla distanza di Mahalanobis. Scatterplot PalletLoad - EffDuration con TruckType T34, magazzino DSE_BG03

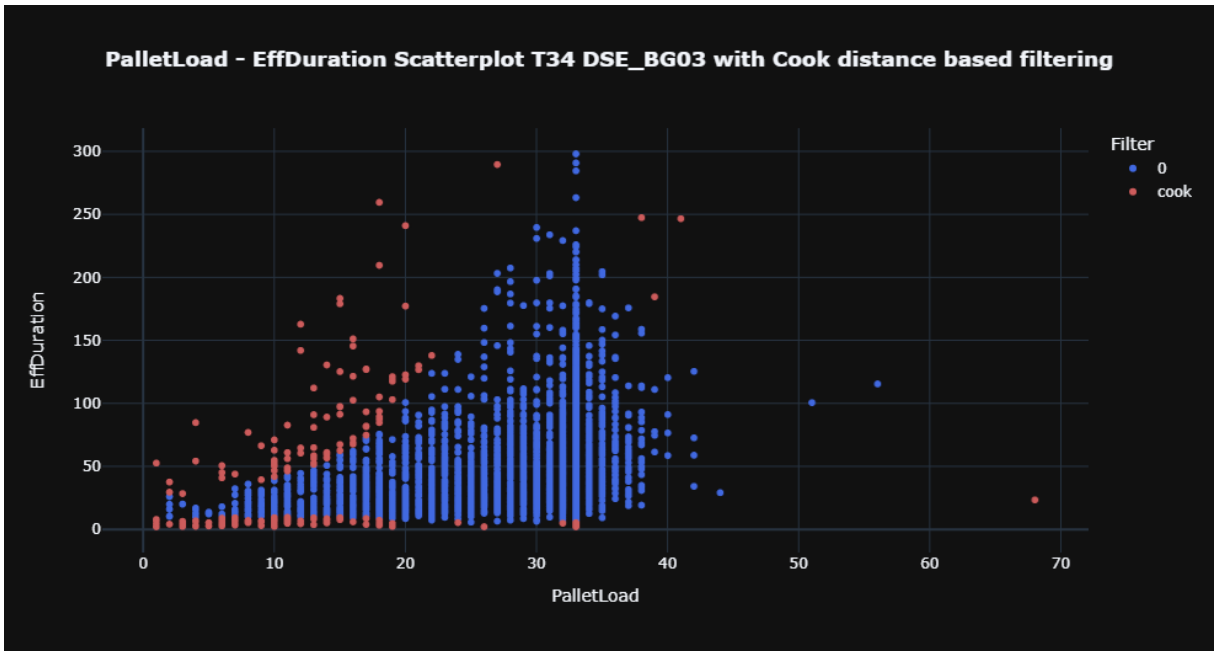

Figura 16 - Filtraggio basato sulla distanza di Cook. Scatterplot PalletLoad - EffDuration con TruckType T34, magazzino DSE_BG03

Dunque, come spiegato, in questo particolare dominio tali tecniche di individuazione degli outlier si rivelano poco efficaci.

Riprendendo il ragionamento precedente, per cui, in questo contesto, più un valore è lontano dalla retta di regressione, più probabilmente è determinato da un errore umano, si può pensare di automatizzare un processo di determinazione di una distanza dalla retta di regressione oltre la quale un valore è considerato outlier.

Una tecnica potrebbe essere quella di utilizzare, come tale distanza, un multiplo della deviazione standard (σ) dei valori sull'asse y (target). Tuttavia, se la distribuzione dei valori non segue la curva normale, la deviazione standard perde di significato statistico.

Nel nostro contesto i valori sull'asse y (target) seguono praticamente sempre una distribuzione riconducibile a una curva normale mediante trasformazioni (tipicamente lognormale [40]). Dunque è possibile trasformare i valori del target (durata effettiva) al fine di ottenere una distribuzione normale, per poi operare sui dati trasformati, in modo da rendere statisticamente significativa la deviazione standard. Inoltre, per valori distribuiti secondo una curva normale è attuabile la regola 68-95-99.7 [19], per la quale circa il 95% dei valori è a distanza minore di 2σ.

Seguendo questo ragionamento si può pensare di effettuare una trasformazione Box-Cox [17] dei valori del target, in modo da ottenere una distribuzione normale, e poi calcolare, per i dati trasformati, la retta di regressione [20] e considerare outlier i punti a distanza maggiore di 2σ da essa. Infine, per mantenere dati realistici e interpretabili, invertire la trasformazione in modo da riottenere i valori del target originali

Il procedimento può essere schematizzato come segue:

- Si effettua una trasformazione Box-Cox dei valori del target, sfruttando la rilevazione automatica del parametro λ offerta dalla apposita funzione della libreria SciPy [18]. Il passaggio da Figura 17 a Figura 18 mostra il risultato di tale trasformazione: è possibile osservare come i valori sull'asse y passino dal seguire una distribuzione lognormale (Figura 17) a seguire una distribuzione normale (Figura 18)
- Si calcola, per i valori della durata effettiva trasformati, la retta di regressione
- Si considerano outlier i punti a distanza maggiore di 2σ da tale retta di regressione, come mostrato in Figura 19. Come si vede, vengono considerati verosimili la maggior parte dei valori
- Si inverte la trasformazione per riottenere i valori del target nel dominio originale. In Figura 20 viene mostrato il risultato: come è possibile osservare i valori considerati outlier (pochi rispetto al totale) sono quelli che si trovano a una certa distanza dalla retta di regressione.

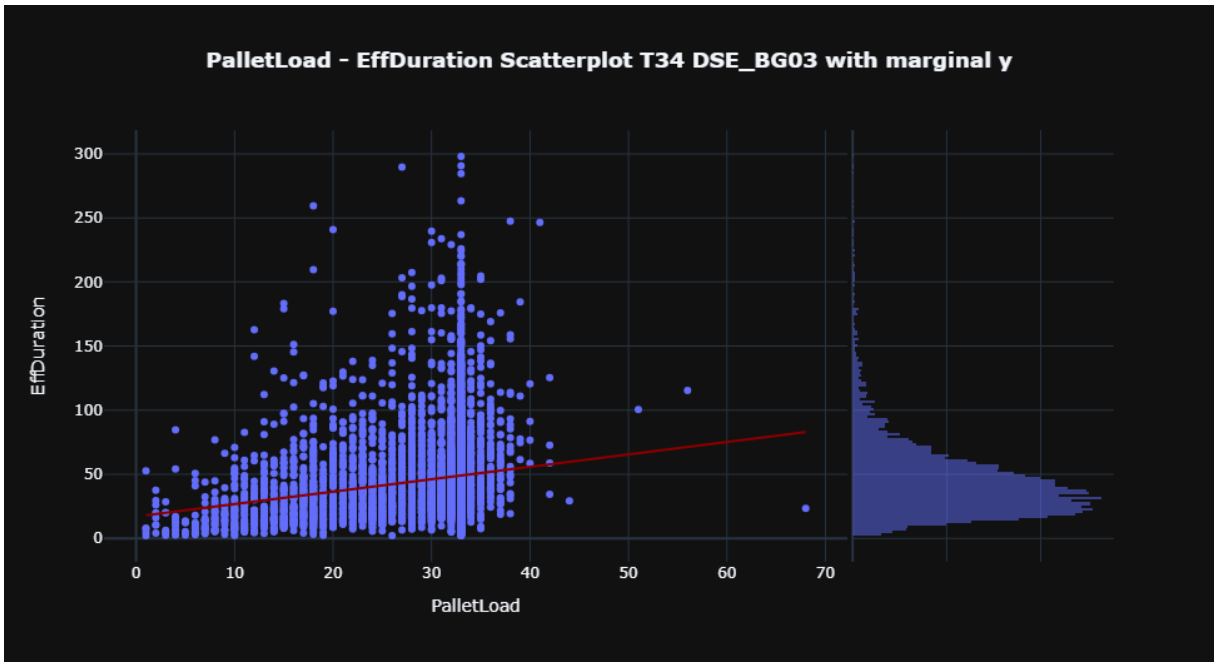

Figura 17 - Scatterplot PalletLoad - EffDuration con TruckType T34 nel magazzino DSE_BG03 + Istogramma distribuzione EffDuration

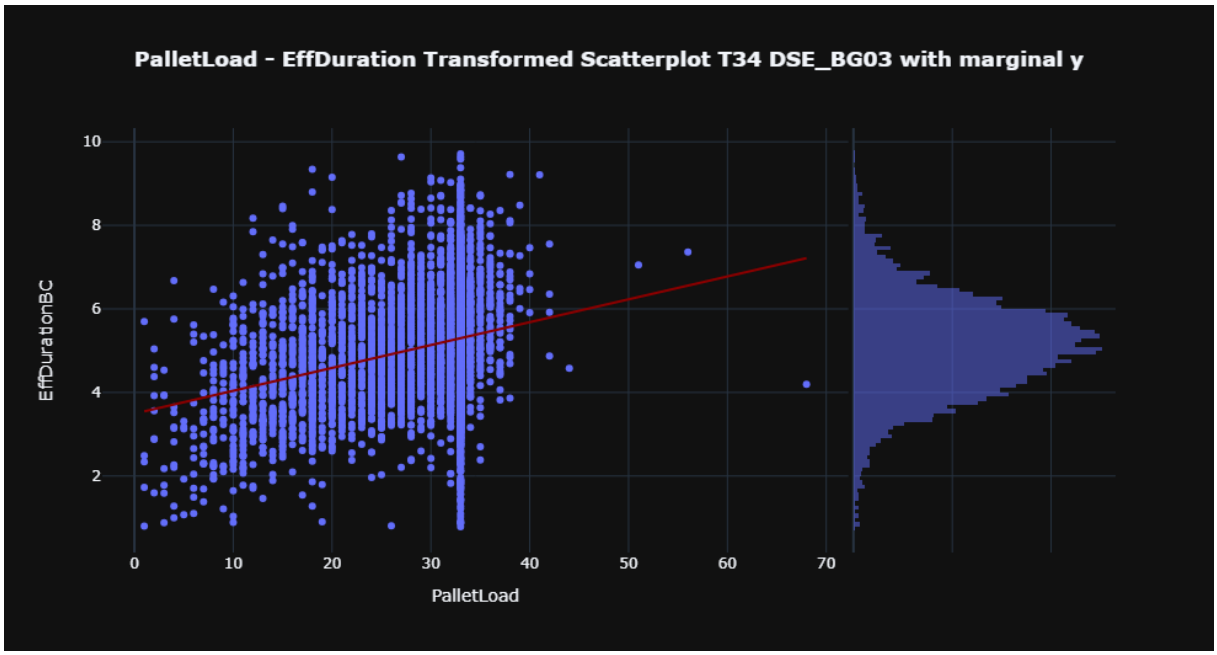

Figura 18 - Scatterplot PalletLoad - EffDuration post trasformazione Box-Cox con TruckType T34 nel magazzino DSE_BG03 + Istogramma distribuzione EffDuration post trasformazione Box-Cox

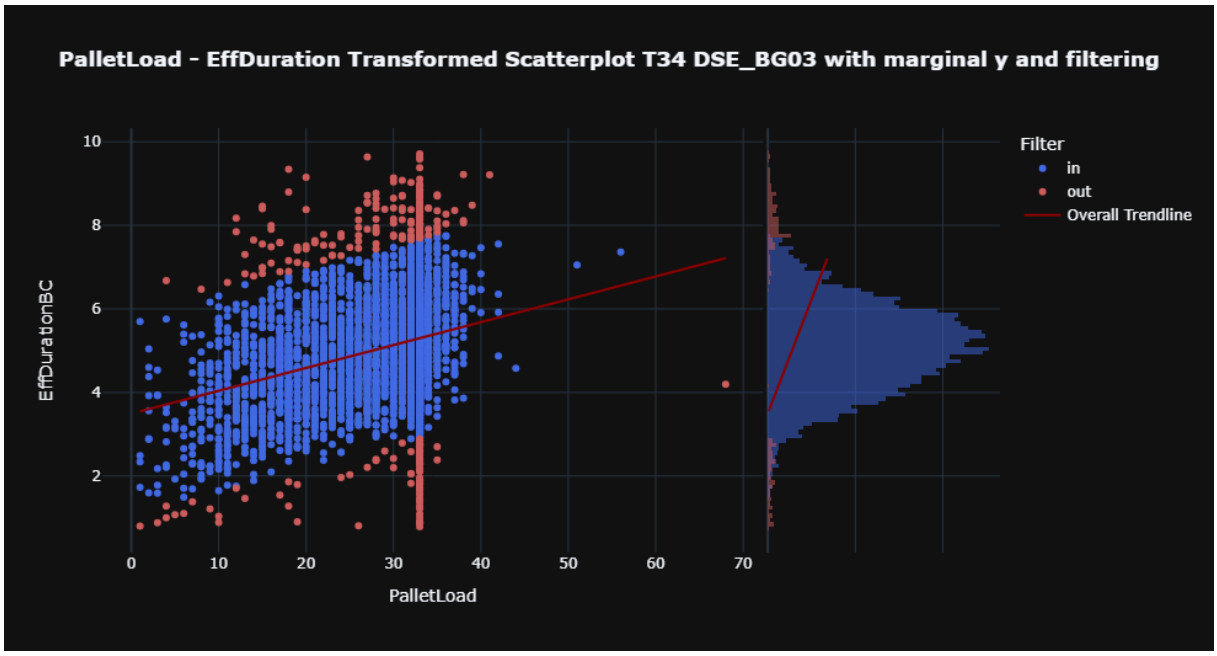

Figura 19 - come Figura 18, ma con gli outlier selezionati dal metodo evidenziati

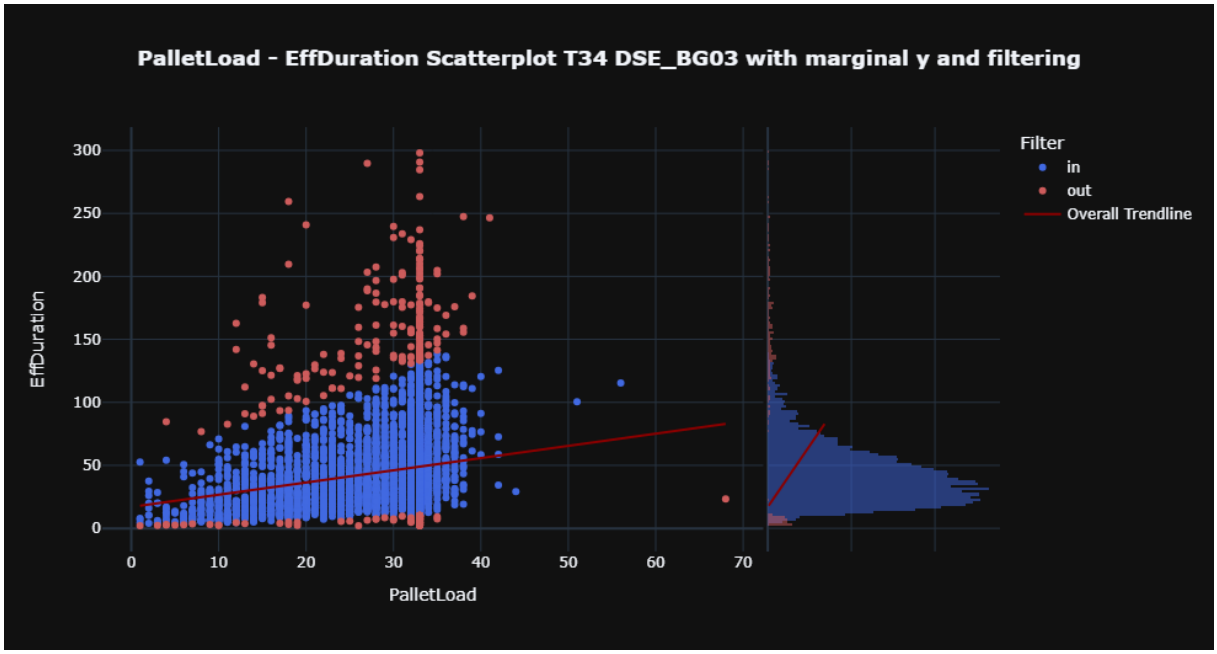

Figura 20 - Outlier selezionati dal metodo evidenziati nel dominio reale + Istogramma distribuzione effDuration

Questo metodo si rivela molto efficiente e molto adattabile a distribuzioni differenti, oltre che interpretabile. In Figura 21 e Figura 22 vengono mostrati due esempi di applicazione: si può vedere come, in entrambi i casi, la maggioranza dei valori sia considerata verosimile e gli outlier siano identificati come i valori che si trovano a una certa distanza dalla retta di regressione.

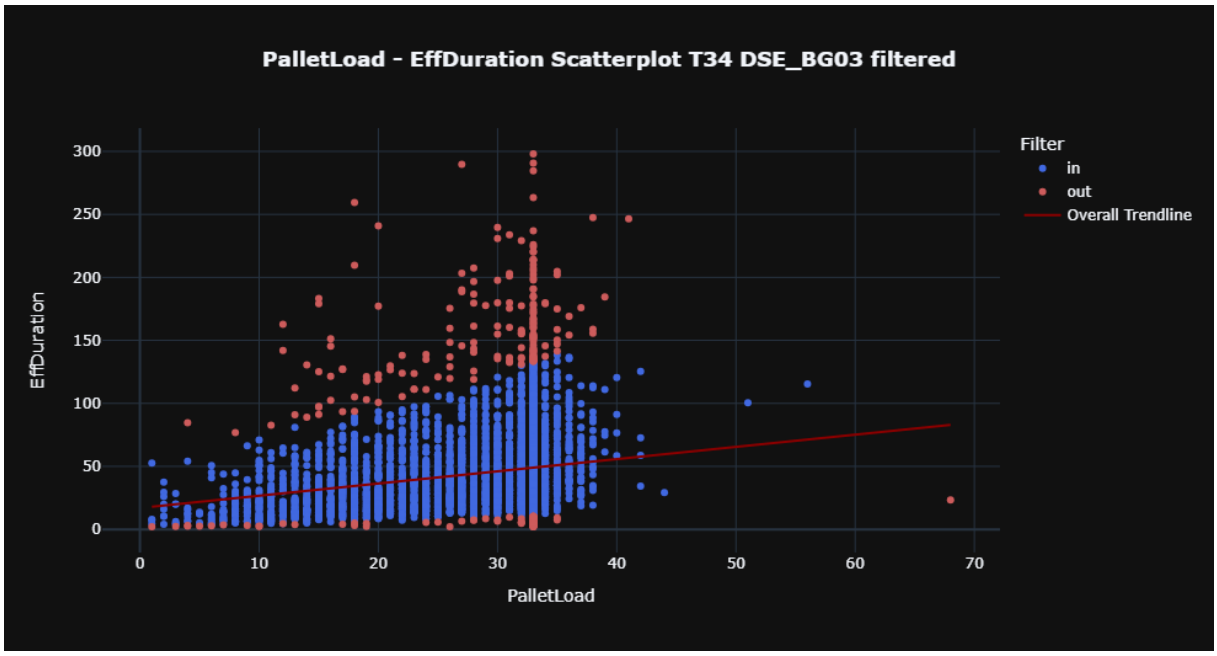

Figura 21 - Scatterplot PalletLoad - EffDuration con TruckType T34 nel magazzino DSE_BG03. Outlier identificati dal metodo evidenziati

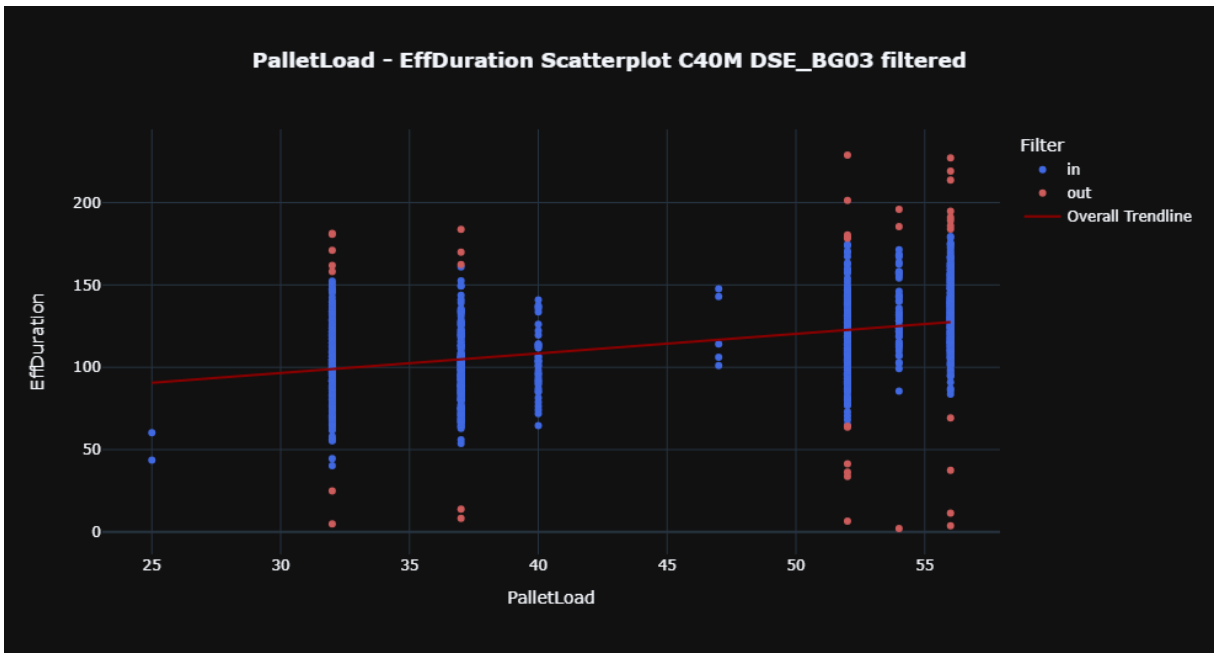

Figura 22 - Scatterplot PalletLoad - EffDuration con TruckType C40M nel magazzino DSE_BG03. Outlier identificati dal metodo evidenziati

Vengono dunque eliminati gli outlier, così identificati, per ogni categoria di TruckType e feature item-azione.

Questo metodo, così formulato, è però utilizzabile solamente nel caso di prenotazioni mono-item. Nel caso di prenotazioni multi-item, infatti, come precedentemente spiegato, non ha significato la relazione tra una feature item-azione e la durata effettiva di un'operazione. Occorre dunque affinare ulteriormente l'approccio, così da renderlo applicabile al caso più generale.

2.2.3. Filtraggio nel caso di prenotazioni multi-item

Nel caso di prenotazioni multi-item, occorre evidentemente identificare un modo per "aggregare" le diverse feature item-azione in un unico indicatore significativo, su cui poi effettuare le elaborazioni

Un modo per farlo può essere quello di considerare l'impatto relativo di una data tipologia item-azione sulla durata dell'operazione, attribuendo a ogni tipologia item-azione un "peso" sulla base della conoscenza del dominio: ad esempio, se per caricare un pallet occorre un tempo mediamente doppio del tempo che si impiega per scaricarne uno, si potranno ipotizzare pesi relativi rispettivamente di 1 per la tipologia pallet-carico e di 0.5 per la tipologia pallet-scarico. Si potrà quindi sintetizzare un indicatore come media ponderata, o formula analoga.

Sulla base di ciò, il procedimento precedentemente definito per il caso mono-item può essere così adattato:

- A ogni tipologia item-azione viene assegnato un peso (valore compreso tra 0 e 1, personalizzato per ogni magazzino)
- Si moltiplica il valore di ogni feature item-azione che compone l'operazione, per il relativo peso e si sommano i valori ottenuti. In questo modo si ottiene, per ogni prenotazione multi-item, un parametro che permetta di trattare l'operazione come mono-item. Questo è necessario in quanto cerchiamo una correlazione bidimensionale tra la durata effettiva dell'operazione e le azioni (per i relativi item) che la compongono
- Si utilizza la somma pesata dei valori delle differenti feature item-azione, che compongono l'operazione multi-item, per poter procedere con il metodo spiegato nel Sottocapitolo 2.2.2. La somma pesata funge da feature item-azione di un'operazione mono-item.

3. Modelli considerati

La fase di *Data Preparation* (*Feature Selection* + *Data Cleaning*) fornisce un Dataset, con dati meno sporchi, composto da:

- 9 feature categoriche: Hour, DayOfWeek, Month, TruckType, Action, MultiItem, Sender, Receiver, Carrier (per i dettagli si veda il Sottocapitolo 2.1)
- Un numero variabile (personalizzato per ogni magazzino), tipicamente minore o uguale a 4, di feature numeriche non categoriche: le feature item-azione (e.g. PalletLoad, PalletUnload, BoxLoad, BoxUnload)
- La colonna target (valore numerico continuo) che si vuole imparare a stimare a priori, ovvero la durata effettiva di un'operazione.

La dimensione del Dataset, ovvero il numero dei campioni (numero di operazioni evase), varia chiaramente da magazzino a magazzino. Solitamente i Dataset in questione hanno dimensioni medie, con un numero di campioni poco inferiore a 10000.

Essendo la colonna target un valore numerico continuo, la stima di tale valore è un problema di regressione.

Prima di introdurre i modelli di Machine Learning considerati occorre specificare che la maggior parte di questi richiedono input numerici. Dunque, in tal caso, è necessario trattare adeguatamente le feature categoriche per convertirle in forma numerica: questa operazione è chiamata *encoding* [33].

In questo caso in particolare è stata utilizzata una tecnica di encoding denominata Leave-One-Out encoding [34]: per ogni campione si sostituisce il valore di una certa feature categorica con la media dei valori del target (durata effettiva della prenotazione) di tutti gli altri campioni (il corrente escluso) aventi lo stesso valore della data feature categorica (facenti parte della stessa categoria). Ad esempio consideriamo un campione che abbia come valore della feature TruckType "T34": tale valore viene sostituito dalla media (valore numerico) dei valori del target (durata effettiva) di tutti gli altri (il campione considerato è escluso) campioni aventi "T34" come valore di TruckType.

Dei molti modelli di Machine Learning disponibili, applicabili a problemi di regressione, ne sono stati selezionati alcuni, considerati potenzialmente adatti al contesto specifico e alla struttura del Dataset.

In particolare sono stati considerati (e testati) 8 modelli differenti, di diverse tipologie:

- Modelli di regressione lineare: Ridge [21], Lasso [22], ElasticNet [23]
- Random Forest Regressor [24]
- Support Vector Regressor [25]
- Algoritmi di Boosting: LGBMRegressor [26], XGBRegressor [27], CatBoostRegressor [28].

Non è stato preso in considerazione l'utilizzo di reti neurali [29], poiché solitamente richiedono Dataset con un numero di campioni molto alto per evitare l'*overfitting* [30], ovvero la tendenza di un modello di adattarsi troppo ai dati utilizzati per l'allenamento e generalizzare male sui dati nuovi. Inoltre l'allenamento di reti neurali può risultare computazionalmente intensivo e richiedere risorse hardware potenti.

3.1. Modelli di regressione lineare

La regressione lineare è una tecnica di apprendimento automatico utilizzata per modellare la relazione tra una variabile dipendente continua (target: nel nostro caso la durata effettiva) e una o più variabili indipendenti (le feature).

Nei modelli di regressione lineare si assume che la relazione tra le variabili sia approssimativamente lineare. Questi modelli cercano di trovare la migliore retta di regressione che minimizzi la somma dei quadrati delle differenze tra i valori osservati e quelli predetti dal modello.

I modelli di regressione lineare si ritengono potenzialmente adatti al contesto in esame poiché la maggior parte dei magazzini esegue principalmente operazioni mono-item, nelle quali effettivamente la relazione tra le feature item-azione e la durata dell'operazione è per sua natura approssimativamente lineare.

Si considerano i tre modelli di regressione lineare principali: Ridge [21], Lasso [22] ed ElasticNet [23].

3.1.1. Ridge Regression

La regressione Ridge è una variante della regressione lineare che introduce una penalizzazione (regolarizzazione) L2 [31] ai coefficienti del modello. Questo aiuta a prevenire il fenomeno di overfitting, in particolare quando le feature sono fortemente correlate.

3.1.2. Lasso Regression

La regressione Lasso è simile a Ridge, ma introduce una penalizzazione (regolarizzazione) L1 [31], la quale ha l'effetto ulteriore di portare alcuni coefficienti a zero, fungendo così anche da metodo di selezione delle feature.

3.1.3. ElasticNet

ElasticNet è una combinazione di Ridge e Lasso: sono introdotte entrambe le penalizzazioni. Questo modello è utile quando ci sono feature fortemente correlate e molte di esse sono poco informative.

3.2. Random Forest Regressor

Il Random Forest Regressor [24] è un modello basato su alberi decisionali. Per stimare il valore della variabile di output utilizza una "foresta" casuale costituita da un insieme di alberi decisionali, ognuno dei quali contribuisce con una previsione.

Questo modello è robusto, in grado di gestire complessità nei dati e catturare relazioni non lineari tra le variabili.

È potenzialmente adatto al contesto in questione perché, avendo molte feature categoriche, la capacità di catturare relazioni non lineari tra le variabili potrebbe rivelarsi utile nella stima.

3.3. Support Vector Regressor

Il Support Vector Regressor (SVR) [25] è un modello che estende il concetto di Support Vector Machine (SVM) [32] a problemi di regressione. Mappando i dati in uno spazio delle feature di dimensioni superiori, SVR risulta utile quando si cercano stime accurate in presenza di relazioni complesse nei dati.

Dal momento che, nel nostro caso, il Dataset presenta sia molte feature categoriche che feature numeriche, è possibile supporre che ci siano sia relazioni lineari che non lineari tra le variabili. Questo rende SVR un buon candidato per il contesto.

3.4. Algoritmi di Boosting

Gli algoritmi di boosting sono tecniche di apprendimento automatico che combinano diversi modelli più deboli per creare un modello forte. Tali modelli vengono allenati in sequenza, dando più peso agli errori commessi dai modelli precedenti. Questa tecnica è utile per migliorare la precisione delle previsioni.

Gli algoritmi di boosting sono potenzialmente adatti al contesto grazie alla loro capacità di adattarsi bene a relazioni complesse tra variabili.

3.4.1. XGBRegressor

XGBRegressor (eXtreme Gradient Boosting Regressor) [27] è un algoritmo di boosting che si basa sulla costruzione sequenziale di alberi decisionali. Ogni nuovo albero cerca di correggere gli errori residui degli alberi precedenti, migliorando progressivamente la precisione complessiva del modello.

3.4.2. LGBMRegressor

LGBMRegressor (Light Gradient Boosting Machine Regressor) [26] è un'altra implementazione di boosting basata su alberi decisionali. La sua principale caratteristica distintiva è la velocità sia in fase di allenamento che in fase di previsione.

3.4.3. CatBoostRegressor

CatBoostRegressor (Category Boosting Regressor) [28] è un altro algoritmo di boosting, basato su alberi decisionali, progettato per gestire automaticamente e ottimamente le feature categoriche senza richiedere un encoding specifico. Questo rende CatBoost particolarmente adatto a Dataset con una combinazione di feature categoriche e numeriche.

Nel nostro caso il Dataset è della tipologia appena descritta, dunque CatBoost presumibilmente è, tra i metodi selezionati, uno di quelli con più potenzialità.

3.5. Metriche di valutazione dei modelli

La valutazione delle prestazioni dei modelli di Machine Learning è un passaggio critico nel processo di sviluppo, poiché fornisce informazioni sulla capacità predittiva dei modelli stessi. A tal fine, è essenziale utilizzare metriche di valutazione adeguate che riflettano accuratamente la performance del modello rispetto all'obiettivo della regressione.

In questo contesto, sono state scelte due metriche comuni: il coefficiente di determinazione (R^2) [35] e l'errore assoluto medio (*MAE*) [36].

Il coefficiente di determinazione, spesso indicato come R^2, è una metrica che misura la proporzione della varianza della variabile dipendente (nel nostro caso, la durata effettiva delle operazioni) che può essere spiegata dalle variabili indipendenti (le feature). Il suo valore può variare da 0 a 1, dove 1 indica una previsione perfetta e 0 indica una previsione che non è migliore della semplice media degli output osservati.

R^2 è calcolato mediante la seguente formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

Dove:

- y_i sono i valori osservati della variabile dipendente
- $\hat{y}_i$ sono i valori predetti dal modello
- $\bar{y}_i$ è la media dei valori osservati
- n è il numero totale di campioni

Un valore di R^2 più vicino a 1 indica una migliore capacità del modello di spiegare la variazione nei dati.

L'errore assoluto medio (MAE) è una metrica che misura la media degli errori assoluti tra i valori predetti e quelli osservati. La sua formula è la seguente:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

MAE fornisce una misura della dispersione dei valori predetti attorno ai valori osservati ed è considerato più interpretabile, poiché rappresenta direttamente l'errore medio delle previsioni.

La scelta di R^2 e MAE come metriche di valutazione è motivata dalla necessità di avere una valutazione completa delle prestazioni dei modelli di regressione, considerando sia l'adattamento ai dati che la precisione delle previsioni. L'utilizzo combinato di R^2 e MAE può fornire una visione equilibrata della bontà dei modelli selezionati per il problema in esame.

3.6. Hyperparameter Tuning

Il processo di *Hyperparameter Tuning* [37] è cruciale nell'ottimizzazione delle prestazioni dei modelli di Machine Learning.

Gli iperparametri sono parametri esterni al modello (potenzialmente diversi per ogni modello) che influenzano il suo comportamento durante l'addestramento. A differenza dei parametri interni (coefficienti del modello), gli iperparametri non vengono appresi direttamente dai dati, ma devono essere impostati prima dell'avvio dell'addestramento.

Il tuning degli iperparametri coinvolge la ricerca (all'interno di range di valori specifici per ogni iperparametro considerato) della combinazione ottimale di valori per questi parametri esterni al fine di massimizzare le prestazioni del modello.

Per condurre il tuning degli iperparametri, è stato scelto di utilizzare Optuna [38], un framework di ottimizzazione degli iperparametri open-source. Optuna automatizza il processo di ricerca degli iperparametri ottimali attraverso l'implementazione di algoritmi di ottimizzazione bayesiani. Questo approccio permette di esplorare in modo efficiente lo spazio degli iperparametri, riducendo il numero di iterazioni necessarie per trovare la configurazione ottimale.

Optuna funziona creando uno studio, all'interno del quale vengono eseguite le valutazioni dei modelli con diverse combinazioni di iperparametri. L'obiettivo è trovare la combinazione che massimizza o minimizza una funzione obiettivo, che rappresenta le prestazioni desiderate del modello. Nel nostro caso si sceglie di massimizzare il valore della metrica R^2 (Sottocapitolo 3.5).

Per ciascun modello è stato selezionato un sottoinsieme dei relativi iperparametri da ottimizzare, in particolare:

- Modelli di Regressione Lineare (Ridge, Lasso, ElasticNet):

    - Alpha (α): Parametro di regolarizzazione che controlla l'intensità della penalizzazione. Maggiori valori di alpha aumentano la forza della regolarizzazione.
- Random Forest Regressor:
    - n_estimators: Numero di alberi decisionali nella foresta
    - max_depth: Massima profondità degli alberi decisionali.
- Support Vector Regressor (SVR):
    - C (Penalty Parameter): Costante di regolarizzazione. Controlla la flessibilità del modello
    - kernel: Definisce il tipo di kernel da utilizzare nella funzione di base del modello SVR. I tipi comuni includono kernel lineare, polinomiale, RBF (Radial Basis Function), e altri.
- Algoritmi di Boosting (XGBRegressor, LGBMRegressor, CatBoostRegressor):
    - n_estimators: Numero di alberi nella sequenza di boosting
    - learning_rate: Tasso di apprendimento che controlla la contribuzione di ciascun albero
    - max_depth: Massima profondità degli alberi nella sequenza di boosting
    - subsample: Frazione di campioni utilizzati per addestrare ciascun albero.

Questi parametri sono soggetti a tuning tramite Optuna al fine di ottenere le migliori prestazioni predittive per ciascun modello considerato. Il processo di Hyperparameter Tuning consente di affinare la configurazione dei modelli, migliorandone la capacità di generalizzazione e adattamento ai dati specifici del problema affrontato.

4. Risultati e considerazioni

I risultati ottenuti si rivelano molto simili nei vari magazzini considerati. Pertanto, al fine di favorire la comprensione, si mostrano i risultati per un magazzino campione: DSE_BG03.

Il Dataset viene suddiviso in *Training Set* [39] e *Test Set* [39]: il 70% dei campioni (scelti randomicamente) del Dataset formano il Training Set, mentre il restante 30% il Test Set. Il primo viene utilizzato per addestrare i vari modelli, il secondo per validarli attraverso le metriche discusse nel Sottocapitolo 3.5. In questo modo la validazione avviene su dati nuovi, mai visti dal modello, il che permette di valutare anche la capacità di generalizzare del modello e l'eventuale presenza di overfitting.

Inizialmente vengono provati i modelli senza effettuare Hyperparameter Tuning (valori degli iperparametri di default).

In Figura 23 sono riportati i valori della metrica R^2, calcolata relativamente al Training Set e al Test Set, per ciascuno dei modelli testati, oltre che per la formula di stima attuale ("oldFormula") descritta nel Sottocapitolo 1.3.

La scelta di calcolare e mostrare la metrica per entrambi i Dataset (Training e Test) mira a evidenziare eventuali casi di overfitting: infatti, un'eventuale differenza significativa tra i due valori per un modello potrebbe indicare il verificarsi di tale fenomeno indesiderato. Come si vede in Figura 23 "RandomForest" e "XGBoost", ma anche, in misura minore, "LGBM", rientrano in questa casistica. Tale ragionamento non si applica alla formula di stima attuale, in quel caso, infatti, non abbiamo due fasi distinte di allenamento e previsione, vengono riportati i due valori distinti di R^2 per una semplice ragione di uniformità.

Come detto precedentemente, è il Test Set ad essere usato per la validazione, quindi, per valutare l'efficienza di un modello è più importante il valore di R^2 relativo al Test Set (colonna rossa in figura).

Figura 23 - Metrica R^2 ottenuta da ogni modello relativamente al Training Set e al Test Set

La prima considerazione da fare è che tutti i modelli considerati, pur senza ottimizzazione degli iperparametri, mostrano risultati migliori rispetto alla formula di stima attuale.

Scendendo più nel dettaglio è possibile notare come i tre modelli di regressione lineare (Ridge, Lasso, ElasticNet) mostrino risultati molto simili: questo potrebbe derivare dal fatto che, probabilmente, le differenti penalizzazioni dei coefficienti introdotte, in questo specifico contesto, modificano solo debolmente la retta di regressione.

Rispetto ai modelli di regressione lineare, SVR mostra risultati peggiori, mentre gli altri modelli risultati migliori. Tra questi CatBoostRegressor, modello particolarmente adatto a Dataset con una combinazione di feature categoriche e numeriche (Sottocapitolo 3.4.3), mostra i risultati migliori in assoluto.

Un'ulteriore considerazione da fare è relativa all'overfitting: RandomForestRegressor, XGBRegressor e LGBMRegressor sembrano soffrire di questo problema, il che potrebbe essere dovuto alla complessità di tali modelli, più dipendenti di altri dall'ottimizzazione degli iperparametri.

4.1. Risultati post Hyperparameter Tuning

Seguendo la metodologia descritta nel Sottocapitolo 3.6 è stata compiuta un'operazione di Hyperparameter Tuning per ogni modello.

In Figura 24 sono riportati, nello stesso modo di Figura 23, i valori della metrica R^2 , calcolata relativamente al Training Set e al Test Set, per ciascuno dei modelli provati, oltre che per la formula di stima attuale. Chiaramente questa volta, a differenza dei risultati mostrati in Figura 23, i modelli hanno come valore dei relativi iperparametri considerati (Sottocapitolo 3.6) quelli ottenuti dalla fase di Hyperparameter Tuning

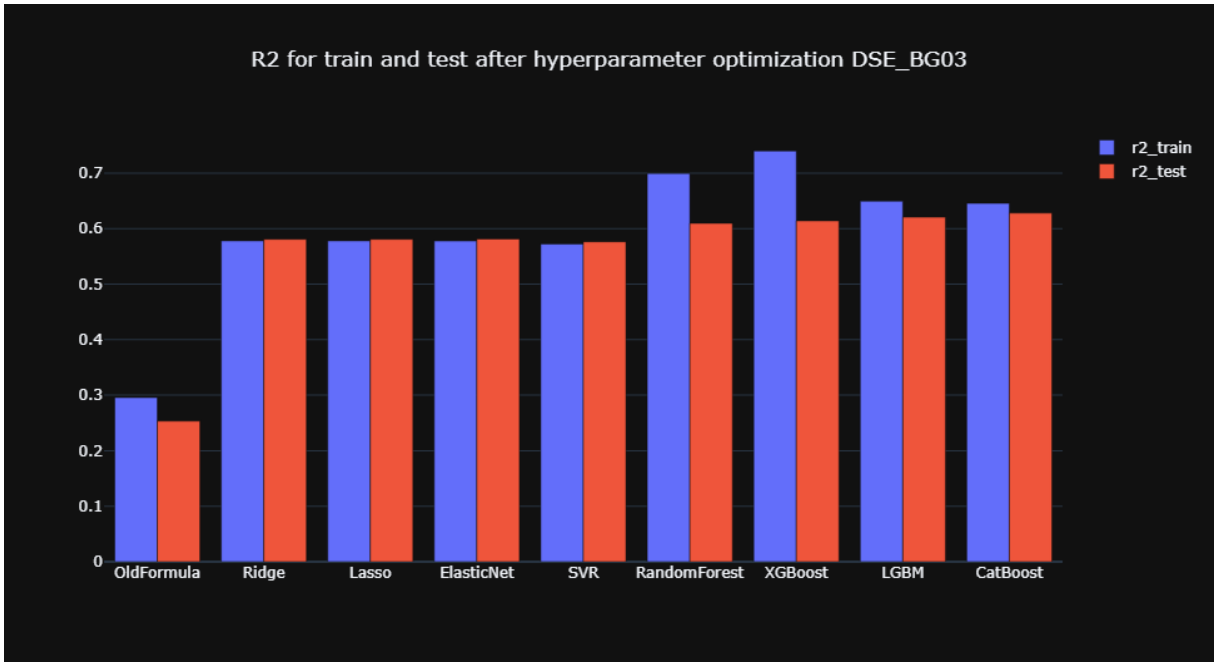

Figura 24 - Metrica R^2 ottenuta da ogni modello relativamente al Training Set e al Test Set post Hyperparameter Tuning

È possibile osservare come tutti i modelli, seppur in misura diversa, migliorino i relativi risultati. In particolare SVR, modello molto sensibile al valore degli iperparametri, mostra un netto miglioramento, arrivando ad avere risultati simili a quelli dei modelli di regressione lineare.

RandomForestRegressor e XGBRegressor mostrano un netto miglioramento per quanto riguarda l'overfitting che però, seppur in misura minore, rimane.

CatBoostRegressor si conferma, anche dopo la fase di Hyperparameter Tuning, il modello che mostra i risultati migliori con un valore della metrica R^2 relativa al Test Set di 0.63.

Per favorire ulteriori analisi dei risultati, in Tabella 2, sono riportati i valori della metrica MAE (Sottocapitolo 3.5) relativa al Test Set per ognuno dei modelli considerati (post Hyperparameter Tuning). I modelli sono riportati in ordine crescente rispetto al valore della metrica MAE che, come spiegato nel Sottocapitolo 3.5, è facilmente interpretabile, poiché rappresenta direttamente l'errore medio delle previsioni: in questo caso l'errore medio, espresso in minuti, che il modello commette nello stimare la durata di un'operazione.

Al fine di comprendere meglio il significato dei valori riportati nella tabella è utile fornire la durata media complessiva di un'operazione nel magazzino campione che stiamo considerando, che si attesta sui 50 minuti.

Modello	**MAE Test Set**
CatBoostRegressor	15.43
LGBMRegressor	15.54
XGBRegressor	15.76
RandomForestRegressor	15.78
SVR	16.04
Ridge	16.59
Lasso	16.60
ElasticNet	16.60
OldFormula	21.13

Tabella 2 - MAE relativa al Test Set per ogni modello post Hyperparameter Tuning. Valori più bassi rappresentano performance migliori

Analizzando i valori riportati in tabella è possibile notare come il miglioramento rispetto alla formula di stima attuale sia netto e significativo. Se si sceglie CatBoostRegressor come modello si ottiene una riduzione nell'errore della stima di 5.7, ovvero di 5 minuti e 42 secondi, pari all'11.4% della durata media di un'operazione nel magazzino campione. Considerando che parliamo della stima della durata di una singola operazione, è facile capire quanto questo valore rappresenti un miglioramento netto: basta pensare all'alto numero di operazioni che vengono effettuate giornalmente in un magazzino logistico.

Come anticipato, i risultati si rivelano molto simili nei vari magazzini considerati: la riduzione dell'errore in percentuale si attesta sempre tra il 10% e il 17%, rispetto alla durata media complessiva di un'operazione nel magazzino.

Per quanto riguarda il tempo necessario ad eseguire una stima, una volta che un modello è allenato, tutti i modelli considerati impiegano meno di un decimo di secondo. L'unico che fa eccezione è SVR, modello tipicamente lento in fase di previsione, che impiega quasi 3 secondi.

Conclusione

Obiettivo di questa tesi era quello indagare come l'applicazione di tecniche di Machine Learning nei modelli previsionali per la pianificazione logistica, a partire da numerosi dati storici, potesse migliorare la stima dei tempi di evasione delle attività di carico e scarico presso magazzini logistici.

In una prima fase il lavoro si è incentrato sulla Data Preparation, in particolare:

- I numerosi dati storici sono stati organizzati in appositi file di tipo "comma separated (.csv)"
- Si è ottenuto un Dataset adatto al dominio del problema attraverso una selezione delle feature basata su parametri statistici
- Mediante un processo di Data Cleaning sono stati rimossi dal Dataset numerosi dati sporchi, dovuti a errore umano nelle rilevazioni. In un primo momento è stata effettuata una selezione dei valori anomali su base monodimensionale, dopodiché, mediante una soluzione ad hoc, su base multidimensionale.

In un secondo momento sono stati selezionati una serie di modelli di Machine Learning considerati adatti al dominio del problema.

È seguita una fase di Hyperparameter Tuning apposita per ognuno dei modelli selezionati attraverso l'utilizzo del framework Optuna.

Infine i modelli sono stati allenati e provati, e i relativi risultati analizzati.

Dai risultati è emerso come l'applicazione di tecniche di Machine Learning riesca effettivamente a migliorare sensibilmente la stima dei tempi di evasione delle attività di carico e scarico presso magazzini logistici. Per tutti i magazzini considerati, infatti, in relazione all'attuale stima, la riduzione dell'errore in percentuale si attesta sempre tra il 10% e il 17%, rispetto alla durata media complessiva di un'operazione nel magazzino.

Questi risultati costituiscono dunque un contributo interessante nell'ambito dell'applicazione di tecniche di Machine Learning nei modelli previsionali per la pianificazione logistica, oltre che una buona base per futuri sviluppi e innovazioni.

Bibliografia

[1] Oltre Solutions. “Load Manager”. https://loadmanager.cloud/. Consultato il 19/01/2024.

[2] Oltre Solutions. “Load Manager Manuale Utente V1.0.3”. Documento interno.

[3] Wikipedia. “Dataset”. https://it.wikipedia.org/wiki/Dataset. Consultato il 19/01/2024.

[4] Nineleaps Technology. "Some Key Machine Learning Definitions". https://medium.com/technology-nineleaps/some-key-machine-learning-definitions-b524eb6cb48. Consultato il 19/01/2024.

[5] Wikipedia. "Box Plot". https://en.wikipedia.org/wiki/Box_plot. Consultato il 19/01/2024.

[6] Wikipedia. "Analysis of Variance". https://en.wikipedia.org/wiki/Analysis_of_variance. Consultato il 19/01/2024.

[7] Wikipedia. "Pearson Correlation Coefficient". https://en.wikipedia.org/wiki/Pearson_correlation_coefficient. Consultato il 19/01/2024.

[8] Wikipedia. "Outlier". https://en.wikipedia.org/wiki/Outlier. Consultato il 19/01/2024.

[9] Wikipedia. "Histogram". https://en.wikipedia.org/wiki/Histogram. Consultato il 19/01/2024.

[10] Wikipedia. "Scatter Plot". https://en.wikipedia.org/wiki/Scatter_plot. Consultato il 19/01/2024.

[11] Wikipedia. "Ordinary Least Squares". https://en.wikipedia.org/wiki/Ordinary_least_squares. Consultato il 19/01/2024.

[12] Wikipedia. "Mahalanobis Distance". https://en.wikipedia.org/wiki/Mahalanobis_distance. Consultato il 19/01/2024.

[13] C.F. Holbert. "Outlier Identification Using Mahalanobis Distance". https://www.cfholbert.com/blog/outlier_mahalanobis_distance/. Consultato il 19/01/2024.

[14] Wikipedia. "Cook's Distance". https://en.wikipedia.org/wiki/Cook%27s_distance. Consultato il 19/01/2024.

[15] Towards Data Science. "Identifying Outliers in Linear Regression - Cook's Distance". https://towardsdatascience.com/identifying-outliers-in-linear-regression-cooks-distance-9e212e9136a. Consultato il 19/01/2024.

[16] Wikipedia. "Multivariate Normal Distribution". https://en.wikipedia.org/wiki/Multivariate_normal_distribution. Consultato il 19/01/2024.

[17] Encyclopedia of Mathematics. "Box-Cox Transformation". https://encyclopediaofmath.org/wiki/Box-Cox_transformation. Consultato il 19/01/2024.

[18] Scipy. "scipy.stats.boxcox". https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.boxcox.html. Consultato il 19/01/2024.

[19] Wikipedia. "Regola 68-95-99,7". https://it.wikipedia.org/wiki/Regola_68-95-99,7. Consultato il 19/01/2024.

[20] Paola Pozzolo. "Analisi dei Residui del modello di Regressione Lineare". https://paolapozzolo.it/analisi-dei-residui-regressione/. Consultato il 19/01/2024.

[21] Scikit-learn. "sklearn.linear_model.Ridge". https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html. Consultato il 19/01/2024.

[22] Scikit-learn. "sklearn.linear_model.Lasso ". https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html. Consultato il 19/01/2024.

[23] Scikit-learn. "sklearn.linear_model.ElasticNet". https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html. Consultato il 19/01/2024.

[24] Scikit-learn. "sklearn.ensemble.RandomForestRegressor". https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. Consultato il 19/01/2024.

[25] Scikit-learn. "sklearn.svm.SVR". https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html. Consultato il 19/01/2024.

[26] LightGBM. "lightgbm.LGBMRegressor". https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html. Consultato il 19/01/2024.

[27] XGBoost. "xgboost.XGBRegressor". https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBRegressor. Consultato il 19/01/2024.

[28] CatBoost. "CatBoostRegressor". https://catboost.ai/en/docs/concepts/python-reference_catboostregressor. Consultato il 19/01/2024.

[29] Wikipedia. "Artificial Neural Network". https://en.wikipedia.org/wiki/Artificial_neural_network. Consultato il 19/01/2024.

[30] Wikipedia. "Overfitting". https://en.wikipedia.org/wiki/Overfitting. Consultato il 19/01/2024.

[31] Diario di un Analista. "Regolarizzazione L1 vs L2 nel Machine Learning: differenze, vantaggi e come applicarle in Python". https://www.diariodiunanalista.it/posts/regolarizzazione-l1-vs-l2-nel-machine-learning-differenze/. Consultato il 19/01/2024.

[32] Wikipedia. "Support Vector Machine". https://en.wikipedia.org/wiki/Support_vector_machine. Consultato il 19/01/2024.

[33] Deniz Gunay. "Feature Encoding". https://medium.com/@denizgunay/feature-encoding-f099a6c1abe8. Consultato il 19/01/2024.

[34] Scikit-learn Contrib. "Leave One Out ". https://contrib.scikit-learn.org/category_encoders/leaveoneout.html. Consultato il 19/01/2024.

[35] Wikipedia. "Coefficient of Determination". https://en.wikipedia.org/wiki/Coefficient_of_determination. Consultato il 19/01/2024.

[36] Wikipedia. "Mean Absolute Error". https://en.wikipedia.org/wiki/Mean_absolute_error. Consultato il 19/01/2024.

[37] Wikipedia. "Hyperparameter Optimization". https://en.wikipedia.org/wiki/Hyperparameter_optimization. Consultato il 19/01/2024.

[38] Optuna. “Optuna – A hyperparameter optimization framework”. https://optuna.org/. Consultato il 19/01/2024.

[39] Wikipedia. "Training, Validation, and Test Data Sets". https://en.wikipedia.org/wiki/Training,_validation,_and_test_data_sets. Consultato il 19/01/2024.

[40] Wikipedia. "Log-Normal Distribution". https://en.wikipedia.org/wiki/Log-normal_distribution. Consultato il 19/01/2024.