Università Politecnica delle Marche
facoltà di ingegneria

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

---



TESI DI LAUREA

Studio e confronto di tecniche per la predizione del tasso di occupazione delle colonnine di ricarica dei veicoli elettrici

Study and Comparison of Techniques for Predicting the Occupancy Rate of Electric Vehicle Charging Stations

Relatore:
Prof. Domenico Potena

Candidato:
Nicola Sebastianelli

Correlatrice:
Dott.ssa Alice Moretti

Anno Accademico 2025–2026

Sommario

La transizione verso la mobilità elettrica e lo sviluppo delle relative infrastrutture di ricarica (EV) rappresentano sfide centrali per la pianificazione delle moderne Smart City. Una gestione efficiente della rete richiede sistemi intelligenti in grado di anticipare la disponibilità delle colonnine, ottimizzando l'allocazione delle risorse e i tempi di attesa per gli utenti. Tuttavia, la previsione diretta dello stato di occupazione si scontra con l'elevata stocasticità del comportamento umano e con la variabilità del fenomeno.

L'obiettivo di questo elaborato di tesi è progettare e strutturare un'architettura predittiva basata su algoritmi di *Machine Learning* e *Time Series Forecasting*, sviluppando la ricerca attraverso due fasi consequenziali supportate da un'Analisi Esplorativa dei Dati (EDA).

Nella prima fase, lo studio si è concentrato sulla previsione del traffico stradale veicolare nella zona di Milano. Sfruttando un dataset storico aggregato in slot da 30 minuti, è stata definita e testata la *baseline* metodologica. Questa fase ha permesso di tarare algoritmi quali Random Forest, Support Vector Machine, Prophet e NeuralProphet su un fenomeno altamente periodico, definendo un protocollo sperimentale basato sulla tecnica della *sliding window* e valutando i costi computazionali tra la fase di addestramento e l'inferenza.

Nella seconda fase, le configurazioni e la conoscenza acquisite sul dominio del traffico sono state traslate sul dominio dell'occupazione delle colonnine EV. A tale scopo, è stata implementata un'infrastruttura su server per l'acquisizione continua di dati reali tramite API, monitorando lo stato delle infrastrutture in diversi punti strategici all'interno della città di Milano.

In sintesi, l'obiettivo finale dell'elaborato è valutare l'applicabilità, l'affidabilità e i limiti di tali modelli predittivi nello stimare l'occupazione delle colonnine, fornendo uno strumento di analisi capace di supportare una gestione più intelligente delle infrastrutture di ricarica e di ottimizzare i tempi di attesa per gli utenti.

Il presente elaborato è articolato in sei capitoli, suddivisi come segue:

- **Nel Capitolo 1** viene introdotto il contesto del problema, esponendo le motivazioni dello studio, gli obiettivi della ricerca e lo stato dell'arte delle tecnologie a supporto della mobilità elettrica.

- **Nel Capitolo 2** viene descritta la fase di raccolta dati, l'Analisi Esplorativa (EDA) e il *pre-processing*. Si illustrano le differenze tra il dataset del traffico e i dati EV acquisiti tramite API
- **Nel Capitolo 3** viene presentata la metodologia. Si descrivono le basi teoriche e viene descritta la motivizione degli algoritmi predittivi selezionati.
- **Nel Capitolo 4** si definisce il protocollo di testing e la fase sperimentale, dettagliando l'implementazione della *sliding window*, l'isolamento del periodo di test e le metriche di valutazione scelte.
- **Nel Capitolo 5** vengono presentati e discussi i risultati. Si confrontano visivamente e analiticamente le previsioni sul traffico e sulle colonnine, e si valutano gli esiti del test di correlazione traffico-colonnine.
- **Nel Capitolo 6**, infine, vengono tratte delle conclusioni a seguito delle analisi svolte e vengono specificati dei possibili sviluppi futuri.

Indice

Sommario **1**

1 Introduzione **9**
1.1 Contesto e motivazioni 9
1.2 Caso di studio e Obiettivi 10
1.3 Metodi per la previsione 10

2 Raccolta Dati, Analisi Esplorativa e Pre-processing **12**
2.1 Acquisizione dati sul traffico stradale 12
2.1.1 Progettazione del software di raccolta 12
2.1.2 Struttura dei dataset e definizione della metrica target 13
2.2 Acquisizione dati sull'occupazione delle colonnine di ricarica 15
2.2.1 Progettazione del software di raccolta e gestione dei limiti API . . . 15
2.2.2 Struttura del dataset e variabile target 17
2.3 Analisi Esplorativa dei Dati (EDA) 17
2.3.1 Dinamiche dei flussi di traffico stradale 18
2.3.2 Dinamiche di occupazione e selezione colonnine EV 22
2.4 Pre-processing e pulizia dei dati 29
2.4.1 Pre-processing del dataset sul traffico e ricostruzione dati 29
2.4.2 Pre-processing del dataset EV e imputazione dati 33

3 Metodologia e Modelli Predittivi **36**
3.1 Approccio alla previsione delle serie storiche 36
3.2 Algoritmi selezionati 37
3.2.1 Modelli Statistici Classici: SARIMA 37
3.2.2 Modelli Additivi e Ibridi: Prophet e NeuralProphet 37
3.2.3 Modelli di Machine Learning e Deep Learning: Random Forest, SVM e LSTM 38
3.3 Note metodologiche sui Large Time Series Models 41

4 Protocollo Sperimentale **42**
4.1 Definizione del protocollo sperimentale 42
4.1.1 Implementazione della *Sliding Window* 42

4.1.2 Isolamento del periodo di test: Scorrimento Giornaliero e Scorrimento a 30 minuti . . . 44
4.2 Impostazione dei parametri per ciascun modello . . . 45
4.3 Metriche di valutazione degli errori . . . 47
4.3.1 Analisi dell'andamento e del decadimento dell'errore . . . 49
4.4 Tracciamento dei costi computazionali e latenza . . . 49
4.4.1 Ambiente software e infrastruttura di calcolo . . . 50

5 Risultati e Discussione **51**
5.1 Risultati sul dataset del traffico stradale . . . 51
5.1.1 Analisi visiva: sovrapposizione curve reali e previsioni . . . 51
5.1.2 Valutazione quantitativa dell'errore (MAE e MAPE) . . . 53
5.1.3 Decadimento dell'errore (orizzonte 48 ore) e confronto sliding window (24h e 30 min) . . . 58
5.2 Risultati sul dataset delle colonnine EV . . . 62
5.2.1 Analisi visiva: sovrapposizione curve reali e previsioni . . . 62
5.2.2 Valutazione quantitativa e analisi di stabilità (media e varianza del MAE sui fold) . . . 64
5.2.3 Impatto della frequenza di sliding window (step 24h e step 30 min) . 70
5.3 Studio di correlazione statistica . . . 71
5.3.1 Analisi di correlazione spazio-temporale . . . 72
5.3.2 Validazione del traffico come predittore indiretto . . . 75

6 Conclusioni **77**
6.1 Sintesi del lavoro e risultati chiave . . . 77
6.2 Contributi originali e valore dell'elaborato . . . 78
6.3 Limiti del lavoro . . . 78
6.4 Sviluppi futuri . . . 79

Bibliografia **80**

Elenco delle figure

2.1 Mappa territoriale delle stazioni di monitoraggio stradale a Milano 18
2.2 Distribuzione statistica della variabile Δv 19
2.3 Andamento temporale delle rilevazioni orarie 20
2.4 Heatmap aggregata della congestione media in funzione del giorno della settimana e dell'ora 21
2.5 Heatmap di congestione del traffico per singola stazione di monitoraggio . . 22
2.6 Tasso di affidabilità delle API (assenza di errori) aggregato per singola zona di monitoraggio 23
2.7 Calcolo dello Score Informativo per la selezione delle colonnine 23
2.8 Andamento temporale dell'occupazione per singola stazione 24
2.9 Confronto medio tra prese totali installate, prese libere e prese attivamente occupate per zona 25
2.10 Distribuzione statistica e dispersione del tasso di occupazione per singola zona 26
2.11 Tasso di saturazione per zona, inteso come la percentuale di tempo in cui l'infrastruttura risulta completamente occupata (zero prese libere) 26
2.12 Andamento del tasso di occupazione medio aggregato lungo i giorni della settimana 27
2.13 Trend orario del tasso di occupazione medio aggregato per l'intera infrastruttura di ricarica 27
2.14 Dettaglio del trend orario di occupazione per singola zona. 28
2.15 Heatmap del tasso di occupazione medio generale in funzione del giorno e dell'ora. 28
2.16 Validazione visiva e distributiva Stazione 1 (Bocconi) 31
2.17 Analisi di coerenza di ricostruzione Stazione 1 (Bocconi) 32
2.18 Validazione visiva e distributiva Stazione 5 (Centrale 1) 32
2.19 Dashboard statistico Stazione 5 (Centrale 1) 33
2.20 Validazione visiva post-ricostruzione 35

3.1 Architettura logica e flusso dell'informazione all'interno di una cella LSTM 40

5.1 Sovrapposizione delle previsioni dei modelli sul dato reale (Δv) per le singole stazioni di monitoraggio (Scorrimento a 1 giorno) 52

5.2 Sovrapposizione delle previsioni dei modelli sul dato reale (Δv) per le singole stazioni di monitoraggio (Scorrimento a 30 minuti) 53
5.3 Confronto sulle metriche MAE (sinistra) e MAPE (destra) aggregate sull'intera rete (Scorrimento 1 giorno) 54
5.4 Errore Medio Assoluto (MAE) per singola stazione di monitoraggio (Scorrimento 1 giorno) 55
5.5 Errore Percentuale (MAPE %) per singola stazione di monitoraggio (Scorrimento 1 giorno) 55
5.6 Confronto sulle metriche MAE (sinistra) e MAPE (destra) aggregate sull'intera rete (Scorrimento 30 minuti) 56
5.7 Errore Medio Assoluto (MAE) per singola stazione di monitoraggio (Scorrimento 30 minuti) 56
5.8 Errore Percentuale (MAPE %) per singola stazione di monitoraggio (Scorrimento 30 minuti) 56
5.9 Andamento temporale del MAE aggregato a livello di rete (Scorrimento 1 giorno) 57
5.10 Andamento temporale del MAPE aggregato a livello di rete (Scorrimento 1 giorno) 57
5.11 Andamento temporale del MAE aggregato a livello di rete (Scorrimento 30 minuti). 58
5.12 Andamento temporale del MAPE aggregato a livello di rete (Scorrimento 30 minuti). 58
5.13 Grafici di decadimento dell'errore (MAE) all'aumentare dell'orizzonte di previsione (da 0 a 48 ore) per le singole stazioni (Scorrimento 1 giorno) . . 59
5.14 Grafici di decadimento dell'errore (MAE) all'aumentare dell'orizzonte di previsione (da 0 a 48 ore) per le singole stazioni (Scorrimento 30 minuti) . . 60
5.15 Costi computazionali dei modelli (Scorrimento 1 giorno): Tempo di addestramento offline in secondi (sinistra) e Latenza di inferenza online in millisecondi per slot (destra) 61
5.16 Costi computazionali dei modelli (Scorrimento 30 minuti): Tempo di addestramento offline in secondi (sinistra) e Latenza di inferenza online in millisecondi per slot (destra) 61
5.17 Grafici delle curve reali e predette per le 7 colonnine EV target (Scorrimento 1 giorno) 63
5.18 Grafici delle curve reali e predette per le 7 colonnine EV target (Scorrimento 30 minuti). 64
5.19 Classifica globale dei modelli predittivi sull'occupazione EV (Scorrimento 1 giorno): confronto sulle metriche MAE (sinistra) e MAPE (destra) aggregate 65
5.20 Errore Medio Assoluto (MAE espresso in % di occupazione) disaggregato per singola colonnina EV target (Scorrimento 1 giorno) 66

5.21 Errore Percentuale (MAPE %) disaggregato per singola colonnina EV target (Scorrimento 1 giorno) 66
5.22 Grafici del decadimento dell'errore (MAE %) sull'occupazione EV per le colonnine target (Scorrimento 1 giorno, media aggregata sui 7 giorni). . . . 67
5.23 Classifica globale dei modelli predittivi sull'occupazione EV (Scorrimento 30 minuti): confronto sulle metriche MAE (sinistra) e MAPE (destra) aggregate 67
5.24 Errore Medio Assoluto (MAE %) disaggregato per singola colonnina EV target (Scorrimento 30 minuti). La colonnina 2 è esclusa dall'analisi. 68
5.25 Errore Percentuale (MAPE %) disaggregato per singola colonnina EV target (Scorrimento 30 minuti). La colonnina 2 è esclusa dall'analisi. 69
5.26 Grafici del decadimento dell'errore (MAE %) sull'occupazione EV per le colonnine target (Scorrimento 30 minuti, media aggregata su 336 finestre) . 69
5.27 Costi computazionali dei modelli sull'occupazione EV (Scorrimento 1 giorno): Tempo di addestramento offline in secondi (sinistra) e Latenza di inferenza online in millisecondi per slot (destra) 70
5.28 Costi computazionali dei modelli sull'occupazione EV (Scorrimento 30 minuti): Tempo di addestramento offline in secondi (sinistra) e Latenza di inferenza online in millisecondi per slot (destra). SARIMA e LSTM sono esclusi. 71
5.29 Heatmap dei coefficienti di correlazione di Pearson e Spearman (Lag 0) per singola stazione 73
5.30 Relazione dispersiva tra traffico e occupazione EV per le stazioni 1, 5 e 8 . . 73
5.31 Correlazione di Spearman tra traffico e occupazione EV disaggregata per contesto (Feriale e Weekend) 74
5.32 Analisi di cross-correlazione di Spearman con *time-lag* da 0 a 3 ore 75

Elenco delle tabelle

2.1 Estratto dei metadati statici (`stazioni.csv`) 14
2.2 Estratto delle serie temporali (`traffico.csv`) 14
2.3 Estratto dei dati salvati relativi allo stato delle colonnine EV 17
2.4 Coordinate geografiche dei 9 nodi di rilevamento del traffico 18
2.5 Statistiche descrittive delle misurazioni stradali. Valori espressi in km/h. . . 19
2.6 Elenco delle 8 colonnine selezionate per la costruzione del dataset target. . . 24

4.1 Riassunto delle configurazioni e degli iperparametri adottati per ciascun algoritmo predittivo. 47
4.2 Ambiente software e infrastruttura di calcolo utilizzati 50

5.1 Sintesi delle metriche di errore globali sul dataset delle colonnine EV (media aggregata sulle 7 colonnine attive per l'esperimento a 1 giorno) 65
5.2 Sintesi delle metriche di errore globali sul dataset delle colonnine EV (media aggregata sulle 7 colonnine attive per l'esperimento a 30 minuti) 68
5.3 Risultati dell'analisi di correlazione sincrona (Lag 0) tra traffico (Δv) e occupazione colonnine. Un P-Value < 0.05 indica significatività statistica . 72

Capitolo 1

Introduzione

Questo capitolo introduce il contesto di riferimento e le motivazioni alla base dell'elaborato. Nelle sezioni seguenti verranno analizzate le principali criticità associate alla gestione dell'infrastruttura di ricarica per la mobilità elettrica, con particolare attenzione alla disponibilità dei punti di ricarica e ai fenomeni di congestione. A partire da tali criticità, verranno quindi definiti gli obiettivi della ricerca, con riferimento al caso di studio della città di Milano. Infine, verrano presentati gli algoritmi predittivi analizzati nel corso della trattazione.

1.1 Contesto e motivazioni

Il settore dei trasporti rappresenta oggi una delle principali fonti di emissioni di gas serra a livello globale. In questo contesto, la transizione verso una mobilità più sostenibile è diventata una priorità assoluta e i veicoli elettrici (EV - *Electric Vehicles*) giocano un ruolo cardine nel processo di transizione ecologica dei centri urbani [1].

Tuttavia, la diffusione su larga scala di tali veicoli risulta ancora fortemente condizionata dalla limitata disponibilità e capillarità dell'infrastruttura di ricarica pubblica. Nonostante i continui investimenti, gli utenti affrontano quotidianamente il problema della *range anxiety* (ansia da autonomia) [2], un limite aggravato dall'incertezza sulla reale accessibilità delle stazioni.

Questa problematica si ripercuote direttamente sulla pianificazione dei viaggi, rivelandosi particolarmente critica per i veicoli che devono rispettare stringenti vincoli temporali (come flotte commerciali o utenti con necessità lavorative). Il tempo di attesa presso una stazione di ricarica, infatti, dipende strettamente dal suo livello di occupazione al momento dell'arrivo; raggiungere una colonnina e trovarla in uso introduce un forte elemento di incertezza, causando inefficienze nei tempi di percorrenza e una fruizione subottimale del servizio [3].

Di conseguenza, può essere utile sviluppare sistemi capaci di prevedere l'occupazione futura delle stazioni, così da mitigare l'imprevedibilità del viaggio. L'era dei Big Data fornisce gli strumenti adatti a questo scopo: le infrastrutture fisiche e i sistemi di naviga-

zione generano quotidianamente enormi moli di dati (log di sessione, flussi veicolari, orari di punta). L'elaborazione di questo volume di dati tramite tecniche di *Machine Learning* permette di fare predizione in modo accurato, superando il monitoraggio puramente reattivo e abilitando una gestione intelligente e proattiva della rete di ricarica [4].

1.2 Caso di studio e Obiettivi

L'obiettivo di questa tesi è progettare e valutare un'architettura software predittiva per stimare la disponibilità delle stazioni di ricarica EV. Il caso di studio si concentra sull'area metropolitana di Milano, scelta per la sua dinamicità e per la densità di infrastrutture legate alla mobilità elettrica. Nello specifico, l'analisi si basa sui dati in tempo reale estratti dalle API di TomTom [5] relativi a 8 zone della città, monitorando sia l'occupazione delle colonnine sia il flusso del traffico stradale adiacente.

Per affrontare il problema in modo strutturato, la ricerca è stata suddivisa in due fasi. Inizialmente, lo studio si è concentrato sulla modellazione dei flussi di traffico. Essendo il traffico un fenomeno caratterizzato da una marcata periodicità, è stato utilizzato come baseline di riferimento per la valutazione degli algoritmi predittivi. In questa fase è stato definito il protocollo di test, basato su finestre temporali scorrevoli (*sliding window*) con un avanzamento dinamico (a 30 minuti e giornaliero), permettendo di configurare con precisione gli algoritmi predittivi.

Successivamente, i modelli ottimizzati sui dati del traffico sono stati applicati al dominio della ricarica degli EV. In questa seconda fase l'obiettivo è stato quello di valutare le prestazioni dei modelli utilizzando direttamente i dati sull'occupazione delle colonnine di ricarica, analizzandone in modo rigoroso i limiti e la reale capacità di generalizzazione di fronte all'intrinseca variabilità del fenomeno.

A fronte di queste limitazioni intrinseche, l'ultima parte della tesi introduce una strada alternativa, l'analisi di correlazione tra le dinamiche del traffico e i picchi di ricarica.

1.3 Metodi per la previsione

La previsione di eventi futuri basata su dati storici ricade nel campo del *Time Series Forecasting*. Negli ultimi anni, la letteratura scientifica ha proposto diverse soluzioni per la previsione dei flussi urbani, evidenziando come i modelli puramente lineari fatichino a gestire dinamiche complesse e non stazionarie [6].

Per condurre le analisi previste dall'elaborato, si è scelto di confrontare un set diversificato di algoritmi, rappresentativi delle principali famiglie metodologiche attuali. L'obiettivo è valutare le prestazioni dei diversi approcci e comprenderne vantaggi e limiti, anche in relazione alle caratteristiche dei dati considerati.

Nello specifico, lo studio analizza modelli statistici classici come SARIMA, efficaci nel catturare stagionalità rigide, affiancati ad algoritmi di *Machine Learning* quali Random Forest e *Support Vector Machine* (SVM), capaci di apprendere relazioni non lineari. Per

intercettare le dipendenze temporali più stratificate, si è fatto ricorso ad architetture di *Deep Learning*, in particolare le reti neurali ricorrenti LSTM (*Long Short-Term Memory*). Infine, sono stati testati modelli additivi e ibridi come Prophet e NeuralProphet, che combinano la scomposizione classica delle serie storiche con moduli di apprendimento profondo. I fondamenti teorici e matematici di tali modelli verranno esplorati in dettaglio nel capitolo metodologico.

Capitolo 2

Raccolta Dati, Analisi Esplorativa e Pre-processing

Questo capitolo illustra le metodologie adottate per la costruzione dei dataset su cui si basa l'intero elaborato. Verranno descritte le logiche di acquisizione delle rilevazioni dei dati sul traffico e dell'occupazione delle infrastrutture di ricarica, esplorando l'architettura software sviluppata per garantire l'affidabilità del monitoraggio. Successivamente, verranno esposti i risultati dell'Analisi Esplorativa dei Dati (EDA) in due sezioni dedicate ai rispettivi domini. Infine, verranno delineate le operazioni di pre-processing necessarie per trasformare i dati grezzi in serie storiche regolari, focalizzandosi sullo sviluppo di algoritmi per la risoluzione dei vuoti temporali.

2.1 Acquisizione dati sul traffico stradale

Il monitoraggio dei flussi veicolari costituisce la prima fase della sperimentazione. L'acquisizione è stata effettuata avvalendosi dei servizi web forniti da TomTom, le cui API (*Traffic Flow Segment*) [5] permettono di interrogare in tempo reale lo stato di una specifica coordinata geografica, ricavando i dati aggregati dei dispositivi di navigazione connessi.

La raccolta dati si è concentrata su 9 nodi stradali dislocati nell'area metropolitana di Milano, selezionati per rappresentare contesti con dinamiche differenti (es. strade urbane, vie di scorrimento veloce e zone commerciali). Il processo di interrogazione è stato automatizzato tramite uno script in linguaggio Python, schedulato per essere eseguito con una frequenza di campionamento pari a 10 minuti, coprendo un orizzonte temporale da marzo 2025 a settembre 2026.

2.1.1 Progettazione del software di raccolta

Dal punto di vista architetturale, lo script di acquisizione è stato progettato per mappare direttamente la risposta dell'endpoint RESTful in un database relazionale (MySQL) ospitato su un server locale. Il database normalizza l'informazione dividendola in due tabelle distinte: un'anagrafica `stazioni` contenente la toponomastica e le coordinate GPS

del nodo di rilevamento, e una tabella `traffico` avente come chiave primaria la tripla (`id_stazione`, `giorno`, `orario`).

Il *Listing* 2.1 mostra un estratto della logica dell'applicativo; l'algoritmo itera sulla lista delle stazioni target, compone dinamicamente l'URL della chiamata passando latitudine e longitudine, ed esegue il parsing del payload JSON restituito dal server, isolando esclusivamente le variabili di interesse prima di eseguire l'inserimento nel database.

```
for i, (nome, _, lat_stazione, lon_stazione, _, _) in enumerate(stazioni, start=1):
    # Composizione dinamica dell'endpoint con coordinate specifiche
    url = f"https://api.tomtom.com/traffic/services/4/flowSegmentData/absolute/15/json?key={API_KEY}&point={lat_stazione},{lon_stazione}&unit=KMPH"

    try:
        response = requests.get(url)
        if response.status_code == 200:
            data = response.json()
            if "flowSegmentData" in data:
                flow_data = data["flowSegmentData"]

                # Parsing delle metriche di velocita dal payload JSON
                current_speed = flow_data.get("currentSpeed", "N/A")
                free_flow_speed = flow_data.get("freeFlowSpeed", "N/A")

                # Generazione dei timestamp locali
                current_day = datetime.now().strftime("%Y-%m-%d")
                current_time = datetime.now().strftime("%H:%M:%S")

                # Inserimento transazionale su database MySQL
                cursor.execute("""
                    INSERT INTO traffico (id_stazione, giorno, orario, velocita_attuale, velocita_senza_traffico)
                    VALUES (%s, %s, %s, %s, %s)
                """, (i, current_day, current_time, current_speed, free_flow_speed))
                conn.commit()
    except requests.exceptions.RequestException as e:
        print(f"Errore nella richiesta API per {nome}: {e}")
```

Listing 2.1: Estratto del software di acquisizione per il parsing della risposta API

2.1.2 Struttura dei dataset e definizione della metrica target

Al termine del periodo di osservazione, il database è stato esportato in due file CSV, mantenendo la separazione logica tra le componenti statiche e dinamiche dell'infrastruttura di monitoraggio.

Il primo file, `stazioni.csv`, funge da dizionario geografico. La Tabella 2.1 ne mostra un estratto rappresentativo.

id	nome	via	lat_stazione	lon_stazione	lat_colonnina	lon_colonnina
1	Bocconi	Via Ferdinando Bocconi, 26, 20136 Milano MI	45.447862	9.190672	45.447854	9.190730
2	Citylife	Via Senofonte, 7, 20145 Milano MI	45.475285	9.157795	45.475260	9.157835

Tabella 2.1: Estratto dei metadati statici (`stazioni.csv`)

Le variabili in esso contenute sono:

- **id**: Chiave primaria autoincrementale del nodo stradale.
- **nome** e **via**: Etichette toponomastiche e indirizzo fisico della postazione.
- **latitudine_stazione** e **longitudine_stazione**: Coordinate esatte fornite alle API per interrogare la stazione di rilevamento del traffico.
- **latitudine_colonnina** e **longitudine_colonnina**: Coordinate esatte della colonnina.

Il secondo file, `traffico.csv` (Tabella 2.2), raccoglie invece i dati a risoluzione 10 minuti, ed è collegato al file precedente tramite la chiave esterna.

id_stazione	giorno	orario	velocita_attuale	velocita_senza_traffico
1	2025-03-29	10:36:34	17	24
1	2025-03-29	10:46:35	18	23
1	2025-03-29	10:56:37	19	24

Tabella 2.2: Estratto delle serie temporali (`traffico.csv`)

Le variabili sono definite come segue:

- **id_stazione**: Identificativo relazionale verso la tabella `stazioni.csv`.
- **giorno** e **orario**: *Timestamp* esatto (`YYYY-MM-DD` e `HH:MM:SS`) della risposta del server.
- **velocita_attuale** (v_{reale}): Velocità media dei veicoli in transito, espressa in km/h.
- **velocita_senza_traffico** (v_{ideale}): Velocità di riferimento del tratto stradale, calcolata teoricamente in condizioni di scorrimento libero (*free-flow speed*).

Per gli scopi predittivi dell'elaborato, la variabile di interesse primaria non è la velocità assoluta. Prevedere la semplice velocità di un veicolo non fornirebbe una percezione univoca della congestione, poiché tale metrica è intrinsecamente vincolata ai limiti legali e strutturali della singola via. Al fine di garantire l'astrazione e la generalizzazione del fenomeno, è stata derivata analiticamente un'ulteriore grandezza, definita come la differenza di velocità (Δv):

$$\Delta v_t = v_{ideale,t} - v_{reale,t} \tag{2.1}$$

Questa trasformazione matematica costituisce una metrica diretta del livello del traffico: un valore di Δv prossimo allo zero segnala un traffico scorrevole, mentre valori fortemente positivi evidenziano congestioni e ingorghi.

2.2 Acquisizione dati sull'occupazione delle colonnine di ricarica

Oltre al monitoraggio del traffico, è stata implementata una procedura di acquisizione per registrare lo stato delle infrastrutture di ricarica per veicoli elettrici (EV).

La selezione delle stazioni da monitorare ha richiesto una mappatura spaziale preliminare. Tramite i servizi di ricerca di TomTom, è stata eseguita una scansione geolocalizzata impostando come origine le coordinate esatte dei nodi stradali analizzati in precedenza per il traffico, con un raggio di ricerca massimo di 150 metri. Questo approccio ha permesso di individuare le infrastrutture idonee a garantire la coerenza spaziale per il successivo studio di correlazione.

La selezione delle colonnine di ricarica da monitorare è stata vincolata da due fattori operativi. Il primo riguarda il limite di chiamate giornaliere imposto dalle API, che ha reso necessario ridurre il numero di colonnine per poter mantenere una frequenza di campionamento di 15 minuti. Il secondo deriva da una fase di test preliminare, durante la quale diverse infrastrutture hanno mostrato un'elevata instabilità di rete, restituendo spesso errori come *"Nessun connettore trovato"*. Per evitare di raccogliere misurazioni vuote che avrebbero compromesso l'addestramento dei modelli, i nodi inaffidabili sono stati esclusi. L'analisi è stata quindi ristretta a 8 stazioni (una per zona), scelte per la loro stabilità nel garantire dati in maniera costante e la vicinanza al punti selezionati per il dataset sul traffico.

La raccolta dati è proseguita in modo continuativo da aprile a settembre 2026, interrogando l'endpoint REST *Charging Availability* [5]. Nello specifico, la richiesta HTTP GET è stata strutturata secondo il seguente formato:

```
https://api.tomtom.com/search/2/chargingAvailability.json?
chargingAvailability={id_colonnina}&key={API_KEY}
```

2.2.1 Progettazione del software di raccolta e gestione dei limiti API

Mantenere una frequenza di campionamento di 15 minuti ha richiesto una corretta gestione dei limiti di quota (*rate-limit*) imposti dalle API di TomTom. Per garantire la continuità dei dati e prevenire interruzioni, lo script Python è stato dotato di meccanismi di *fault tolerance*.

Il sistema si basa su una rotazione delle chiavi di autenticazione tramite logica *Round-Robin*. L'applicativo inizializza una lista di chiavi API e ne memorizza l'ordine in un file JSON locale. Come illustrato nel Listing 2.2, il software interroga l'endpoint utilizzando la

prima chiave disponibile; se il server restituisce un codice HTTP 403 o 429 (*Quota Exceeded*), la chiave esaurita viene spostata in fondo alla lista. Il file di stato viene aggiornato e la chiamata viene ripetuta con la chiave successiva.

```python
def esegui_chiamata(url_base):
    global stato_api, oggi_str

    # Controllo preventivo per bloccare le chiamate in caso di esaurimento
    if stato_api.get("blocco_giornaliero") == oggi_str:
        return "BLOCCO_TOTALE"

    chiavi = stato_api.get("chiavi", [])
    totale_chiavi = len(chiavi)

    for _ in range(totale_chiavi):
        chiave_attuale = chiavi[0]
        url_completo = f"{url_base}&key={chiave_attuale}"

        try:
            risposta = requests.get(url_completo, timeout=10)

            # Rotazione Round-Robin in caso di chiave esaurita
            if risposta.status_code in [403, 429]:
                chiavi.append(chiavi.pop(0))
                stato_api["chiavi"] = chiavi

                with open(FILE_STATO, "w") as f:
                    json.dump(stato_api, f, indent=4)
                continue

            return risposta

        except Exception as e:
            return None

    # Attivazione del blocco giornaliero se tutte le chiavi falliscono
    stato_api["blocco_giornaliero"] = oggi_str
    with open(FILE_STATO, "w") as f:
        json.dump(stato_api, f, indent=4)

    return "BLOCCO_TOTALE"
```

Listing 2.2: Logica Round-Robin e gestione degli errori per l'interrogazione delle API

Nel caso in cui tutte le chiavi disponibili si esauriscano nella stessa giornata, l'applicativo attiva un blocco temporaneo. Per evitare di generare *missing values* nel database, lo script inserisce comunque dei record con valori azzerati, riportando nel campo diagnostico l'etichetta *"Tutte le chiavi esaurite"*. Questa tecnica mantiene fissa la griglia temporale, delegando la risoluzione del problema dei dati mancanti alla successiva fase di pre-processing.

2.2.2 Struttura del dataset e variabile target

Le rilevazioni sono salvate in un database MySQL diviso in due tabelle. La tabella `colonnine_note` contiene l'anagrafica statica: mappa ogni stazione indicando l'ID del nodo stradale associato, l'ID hardware della colonnina, il gestore energetico, l'indirizzo e la distanza in metri dal punto in cui viene rilevato il traffico.

La tabella `stato_colonnine` memorizza i dati dinamici e viene aggiornata ogni 15 minuti. La Tabella 2.3 riporta un esempio dei record salvati.

id	giorno	orario	id_colonnina	totali	libere	occupate	fuori_servizio	riservate	sconosciute	note_stato
1	2026-04-21	13:15:02	VUN6Vmn62...	8	7	1	0	0	0	OK
2	2026-04-21	13:15:02	bFF9ywyXJ...	4	4	0	0	0	0	OK
3	2026-04-21	13:15:02	KFuZUwMch...	0	0	0	0	0	0	Nessun connettore trovato

Tabella 2.3: Estratto dei dati salvati relativi allo stato delle colonnine EV

Le variabili salvate per ogni rilevazione sono:

- **id**: Identificativo numerico incrementale del record.
- **giorno** e **orario**: Data e ora esatta (`YYYY-MM-DD` e `HH:MM:SS`) in cui è stata eseguita la richiesta API.
- **id_colonnina**: Codice univoco della colonnina fornito dall'operatore, utilizzato come chiave esterna verso la tabella `stato_colonnine`.
- **prese_totali**: Numero totale di prese installate nella stazione.
- **prese_libere**: Numero di prese funzionanti e disponibili per l'uso in quell'istante.
- **prese_occupate**: Numero di prese utilizzate per una ricarica al momento della misurazione.
- **prese_fuori_servizio, riservate, sconosciute**: Contatori che indicano le prese non utilizzabili a causa di guasti hardware, limitazioni contrattuali o assenza di comunicazione con i server.
- **note_stato**: Stringa testuale diagnostica. Riporta *"OK"* quando la chiamata va a buon fine, oppure salva l'errore specifico restituito dal server (es. *"Errore di Rete"*, *"Nessun connettore trovato"*).

2.3 Analisi Esplorativa dei Dati (EDA)

L'Analisi Esplorativa dei Dati (EDA) costituisce una fase fondamentale per comprendere le caratteristiche strutturali dei dataset prima della definizione dei modelli predittivi. Questa indagine quantitativa e visiva è stata condotta per quantificare la varianza del fenomeno, identificare pattern ciclici ricorrenti ed evidenziare la presenza di eventuali anomalie nelle serie storiche acquisite.

L'analisi è presentata separatamente per i due domini monitorati, evidenziandone le specifiche peculiarità.

2.3.1 Dinamiche dei flussi di traffico stradale

L'indagine sul traffico è iniziata con la mappatura spaziale delle stazioni di rilevamento. La Tabella 2.4 elenca i 9 nodi stradali monitorati e le rispettive coordinate geografiche, mentre la Figura 2.1 ne illustra la dislocazione sul territorio metropolitano di Milano. La selezione ha coperto un campione eterogeneo di viabilità, includendo strade urbane, tratti di tangenziale e vie commerciali.

ID	Nome Stazione	Latitudine	Longitudine
1	Bocconi	45.447862	9.190672
2	Citylife	45.475285	9.157795
3	Viale Certosa	45.495426	9.140054
4	Tangenziale Ovest	45.429429	9.080622
5	Stazione Centrale 1	45.484478	9.204951
6	Stazione Centrale 2	45.484552	9.204714
7	Navigli	45.452137	9.169741
8	Corso Buenos Aires	45.476918	9.197998
9	Porta Romana	45.454743	9.205632

Tabella 2.4: Coordinate geografiche dei 9 nodi di rilevamento del traffico

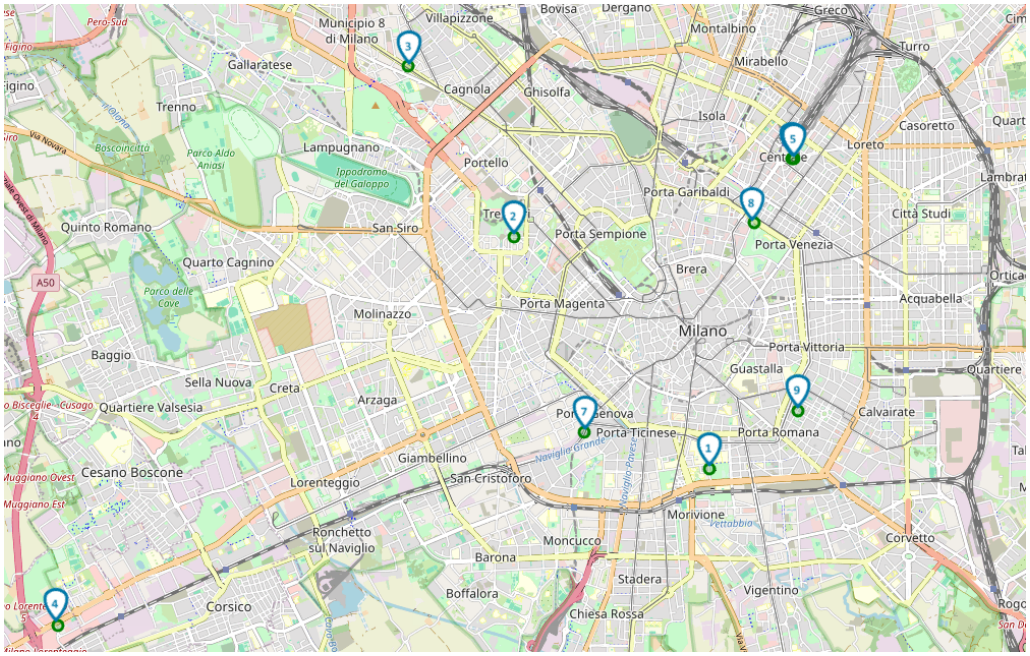


Figura 2.1: Mappa territoriale delle stazioni di monitoraggio stradale a Milano

L'attenzione analitica si è successivamente concentrata sulla variabile target Δv (differenza tra velocità ideale e velocità reale). La Figura 2.2 illustra la distribuzione di

frequenza di questa grandezza. Come prevedibile per un contesto urbano, la distribuzione risulta fortemente asimmetrica a destra: la maggior parte delle rilevazioni si concentra in prossimità dello zero. Ciò indica che la rete viaria è prevalentemente scorrevole o con rallentamenti minimi.

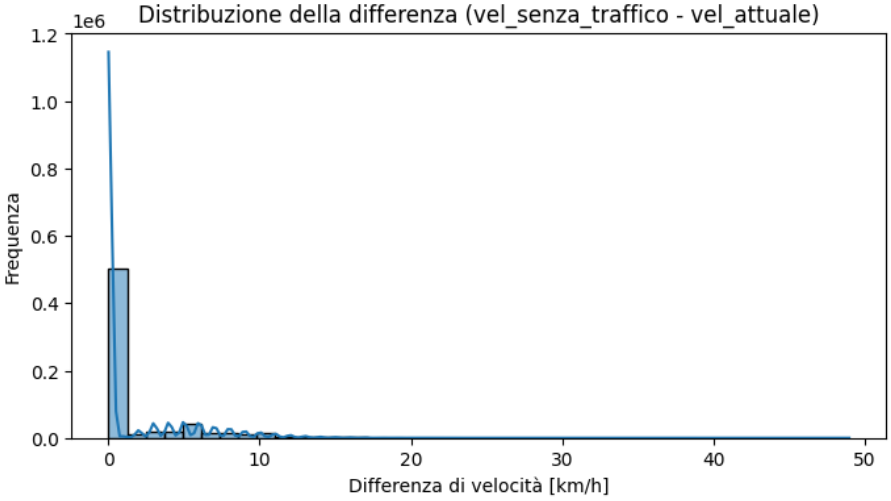


Figura 2.2: Distribuzione statistica della variabile Δv

Le evidenze visive sono confermate dall'analisi quantitativa riportata nella Tabella 2.5. Su un totale di oltre 650.000 campioni, il valore mediano (50° percentile) e il 75° percentile della differenza di velocità sono pari a 0.00 km/h. La congestione rappresenta quindi un evento statisticamente minoritario (la "coda" della distribuzione), con un valore massimo registrato di 49.00 km/h di rallentamento rispetto al limite ideale. Questo sbilanciamento costituisce una sfida strutturale per i modelli di Machine Learning, i quali dovranno essere calibrati per non appiattire la previsione sul valore medio globale (1.48 km/h).

Metrica	**velocita_attuale**	**velocita_senza_traffico**	**differenza (Δv)**
count	653949.00	653949.00	653949.00
mean	21.92	23.40	1.48
std	7.45	7.22	3.10
min	0.00	0.00	0.00
25%	17.00	18.00	0.00
50%	19.00	23.00	0.00
75%	27.00	27.00	0.00
max	56.00	57.00	49.00

Tabella 2.5: Statistiche descrittive delle misurazioni stradali. Valori espressi in km/h.

Il passaggio successivo ha indagato la continuità delle registrazioni sull'asse temporale. La Figura 2.3 scompone l'andamento della variabile Δv per singola stazione.

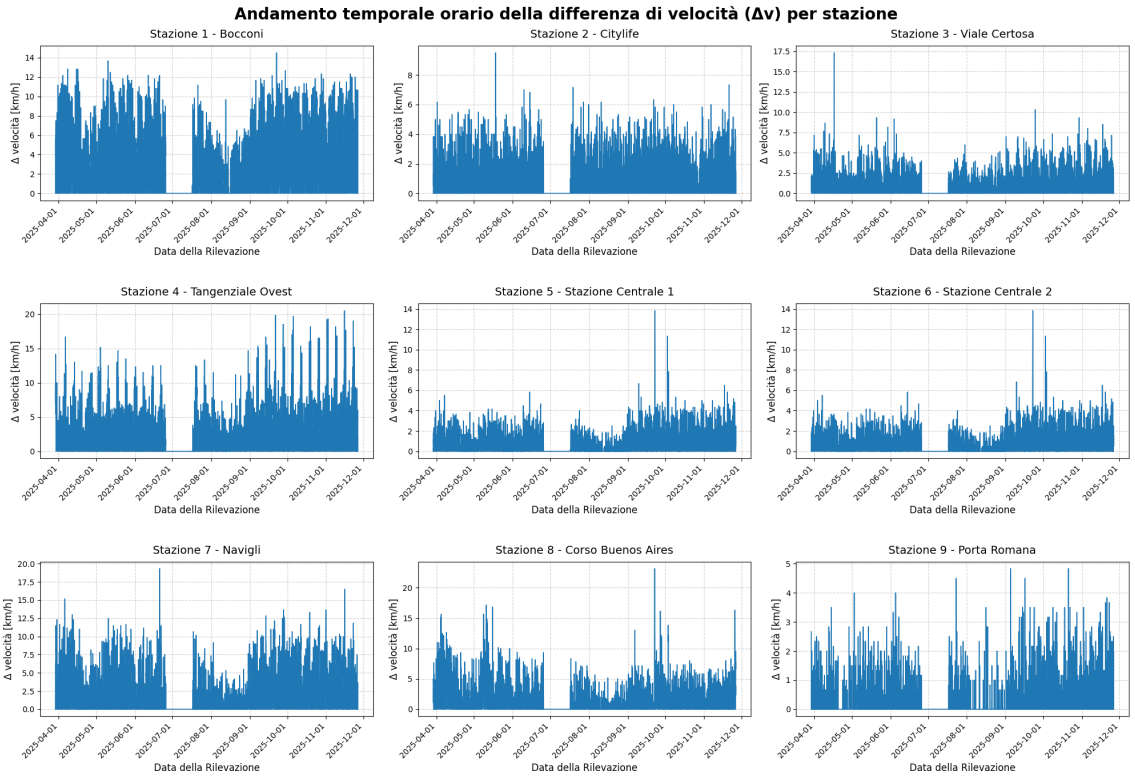


Figura 2.3: Andamento temporale delle rilevazioni orarie

Il grafico evidenzia chiaramente la principale criticità del dataset, la presenza di una marcata discontinuità temporale causata dalla manutenzione del server. Poiché la quasi totalità degli algoritmi per serie storiche (SARIMA, reti neurali ricorrenti) richiede una sequenza di campionamento ininterrotta e costante, l'esplorazione visiva ha confermato la necessità di sviluppare un algoritmo di *gap-filling* e ricostruzione dei dati mancanti in fase di pre-processing.

Infine, nonostante le interruzioni, l'aggregazione dei dati rivela la presenza di una marcata e robusta componente stagionale. La Figura 2.4 illustra l'intensità media del rallentamento organizzata per ora del giorno e giorno della settimana (da 0 per il Lunedì a 6 per la Domenica).

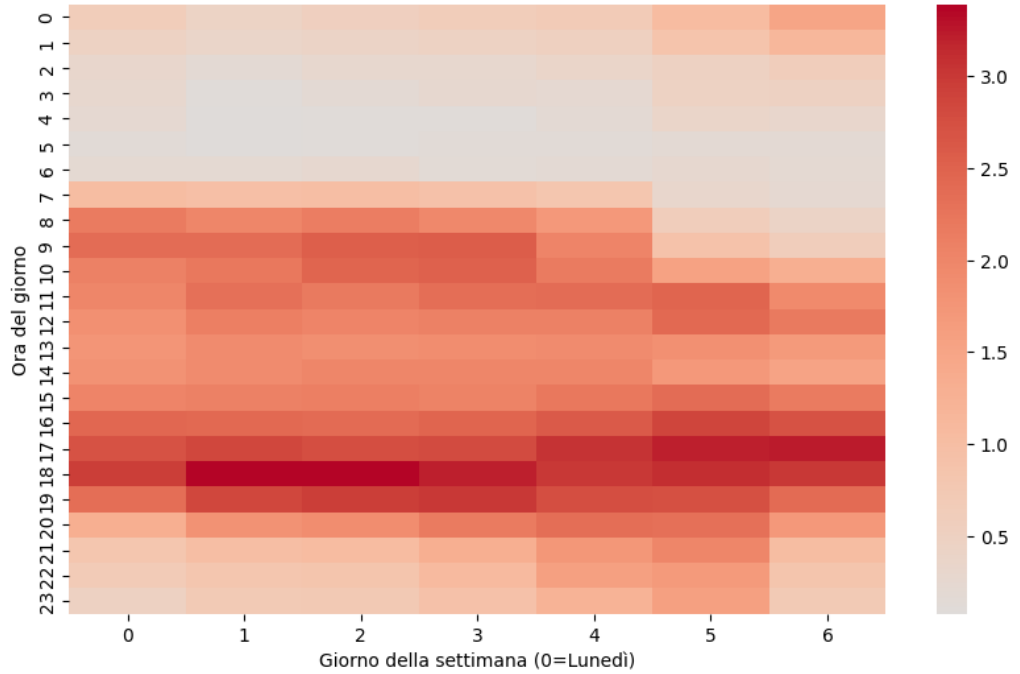


Figura 2.4: Heatmap aggregata della congestione media in funzione del giorno della settimana e dell'ora

L'immagine evidenza un fenomeno ripetitivo. I giorni feriali (da 0 a 4) presentano un andamento con due chiari picchi di congestione legati agli spostamenti pendolari: la fascia mattutina (tra le 07:00 e le 10:00) e quella serale (tra le 17:00 e le 20:00). Durante il fine settimana (5 e 6), il traffico nella mattinata appare notevolmente più scorrevole e diluito.

Tuttavia, nelle heatmap per singola stazione (Figura 2.5), emergono dinamiche locali specifiche. Ad esempio, la Tangenziale Ovest (Stazione 4) concentra la congestione prevalentemente nelle ore centrali e pomeridiane festive, mentre la zona dei Navigli (Stazione 7) evidenzia forti picchi di rallentamento in fascia notturna, concentrati tra il venerdì e il sabato sera in concomitanza con le attività legate allo svago e alla vita notturna cittadina.

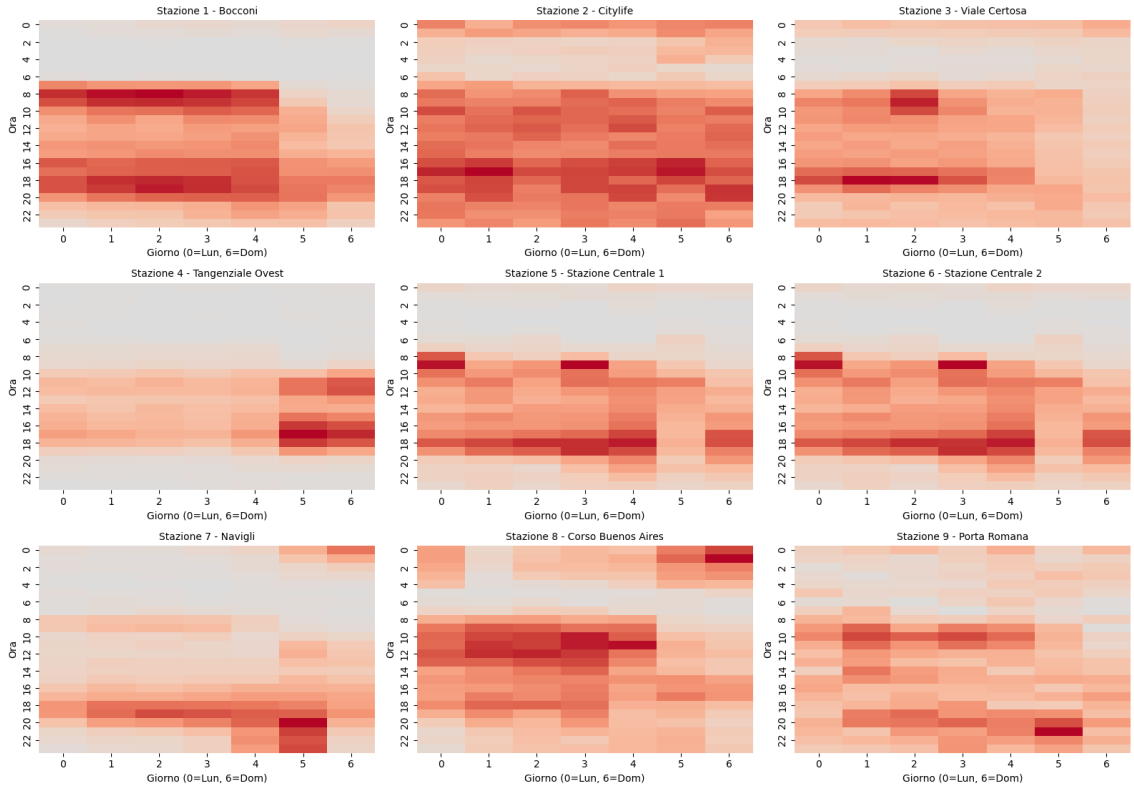


Figura 2.5: Heatmap di congestione del traffico per singola stazione di monitoraggio

Questa spiccata regolarità ciclica, sia aggregata che locale, giustifica matematicamente l'inclusione di variabili temporali esogene (ora, giorno della settimana) nei modelli di Machine Learning e l'adozione di architetture in grado di gestire autocorrelazioni stagionali multiple [7].

2.3.2 Dinamiche di occupazione e selezione colonnine EV

L'esplorazione del dataset relativo all'infrastruttura di ricarica ha evidenziato uno scenario eterogeneo, condizionato da limitazioni hardware e dinamiche stocastiche.

Il primo parametro analizzato è l'affidabilità di rete. Su oltre 179.000 record acquisiti, è stato registrato un tasso di errore globale del 33.41%, causato nella quasi totalità dei casi da mancate risposte del server classificate col codice *"Nessun connettore trovato"*. Come visibile in Figura 2.6, la stabilità varia in modo significativo in base alla posizione. Stazioni come Corso Buenos Aires e Porta Romana presentano un tasso di affidabilità prossimo al 100%, mentre l'infrastruttura in zona Navigli e Citylife evidenzia disservizi frequenti.

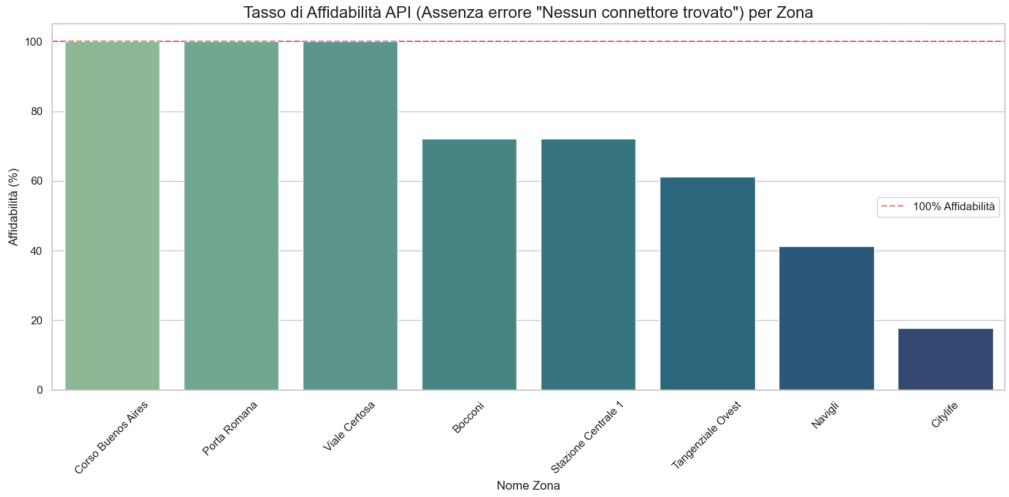


Figura 2.6: Tasso di affidabilità delle API (assenza di errori) aggregato per singola zona di monitoraggio

Il problema dell'affidabilità si interseca con la necessità di selezionare le stazioni ottimali per l'addestramento dei modelli. All'interno del raggio di ricerca di 150 metri da ogni nodo stradale, sono state rilevate molteplici stazioni di ricarica. Per selezionare un'unica colonnina di riferimento per ciascuna zona, è stato formulato uno *Score Informativo* (normalizzato su una scala 0-100).

Questo punteggio valuta la qualità del dato generato dalla colonnina, premiando l'occupazione media, la deviazione standard, la percentuale di dinamismo (frequenza dei cambi di stato) e penalizzando le infrastrutture perennemente vuote o perennemente saturate. La Figura 2.7 mostra i risultati dell'analisi applicata alle colonnine candidate.

```
================================================================================
RISULTATI ANALISI: SCORE INFORMATIVO COLONNINE (Normalizzato 0-100)
================================================================================
Zona           Hardware ID              Provider  Osservazioni  Occ. Media (%)  Dev. Std (%)  Dinamismo (%)  % Vuota  % Piena  Score Informativo
   1 VUN6Vmn62SrL2bTEgNj1LQ                  A2A          3781            23.1          19.8           30.3     26.7      0.0              54.74
   1 g7Q2JIQLUkG-eYyAKw4hcA      Queenergy S.r.l.          3781             6.3           9.9           22.5     62.1      0.0              20.26
   2 bFF9ywyXJxYkCB2-iD_enQ                  A2A          1003             0.0           0.0            0.0    100.0      0.0               0.00
   3 KFuZUwMchLahhkOWC7JWCQ                  A2A          3781            33.0          34.5           25.0     45.0     11.4              78.99
   3 faPJ1tKddtG2eoC7ru4dDA Plenitude On The Road          3781            29.0          27.1           29.6     36.0      2.4              73.31
   4 hjcQ8Y4ew_Q87GS1SbrGBw Plenitude On The Road          3781            16.1          21.7            2.8     58.8      0.1               5.51
   5 1_bXSX1bKKga0akzOrO9qw                  A2A          3781            13.5          16.8           21.9     52.1      0.0              33.69
   5 YFVplryZgGsLaM9sYwfh6w Plenitude On The Road          3781            19.4          21.1           11.3     48.4      0.1              21.87
   7 _lf0j_lBN8bPRGUiuGDGPg                  A2A          3781            15.3          14.2           14.3     27.2      0.0              18.45
   7 _klcPu23awu92alfS6A_AQ                  A2A          3781             9.3          19.5            6.2     80.6      0.2              11.10
   8 -rSGGo3ox0NoCjAAdMt_Nw                  A2A          3781            28.7          32.9           24.9     46.1      8.7              74.82
   8 GsrbHOfA2XlGcpbcL8liew                  A2A          3781            43.0          36.1           21.6     33.1     18.8              71.27
   9 v6t0SuVogqQw8AgNAA2bVA Plenitude On The Road          3781            51.6          36.1           24.5     23.9     25.8              80.92


================================================================================
LE 8 COLONNINE SELEZIONATE PER IL MODELLO:
================================================================================
['VUN6Vmn62SrL2bTEgNj1LQ', 'bFF9ywyXJxYkCB2-iD_enQ', 'KFuZUwMchLahhkOWC7JWCQ', 'hjcQ8Y4ew_Q87GS1SbrGBw', '1_bXSX1bKKga0akzOrO9qw', '_lf0j_lBN8bPRGUiuGDGPg', '-rSGGo3ox0NoCjAAdMt_Nw', 'v6t0SuVogqQw8AgNAA2bVA']
```

Figura 2.7: Calcolo dello Score Informativo per la selezione delle colonnine

Questo ha portato all'isolamento delle 8 colonnine definitive utilizzate per il dataset target. La Tabella 2.6 ne riassume i dati identificativi e il relativo punteggio ottenuto.

Zona	ID Hardware Colonnina	Provider	Score Informativo
1	VUN6Vmn62SrL2bTEgNj1LQ	A2A	54.74
2	bFF9ywyXJxYkCB2-iD_enQ	A2A	0.00
3	KFuZUwMchLahhkOWC7JWCQ	A2A	78.99
4	hjcQ0Y4ew_Q87GS1SbrGBw	Plenitude On The Road	5.51
5	1_bXSX1bKKga0akzOrO9qw	A2A	33.69
7	_lf0j_lBN8bPRGUiuGDGPg	A2A	18.45
8	-rSGGo3ox0NoCjAAdMt_Nw	A2A	74.82
9	v6t0SuVogqQw8AgNAA2bVA	Plenitude On The Road	80.92

Tabella 2.6: Elenco delle 8 colonnine selezionate per la costruzione del dataset target.

Un'ulteriore dimensione di analisi ha riguardato la continuità temporale delle misurazioni acquisite per il sottoinsieme di colonnine definitivo. La Figura 2.8 illustra l'evoluzione cronologica del tasso di occupazione per ciascuna delle 8 colonnine selezionate.

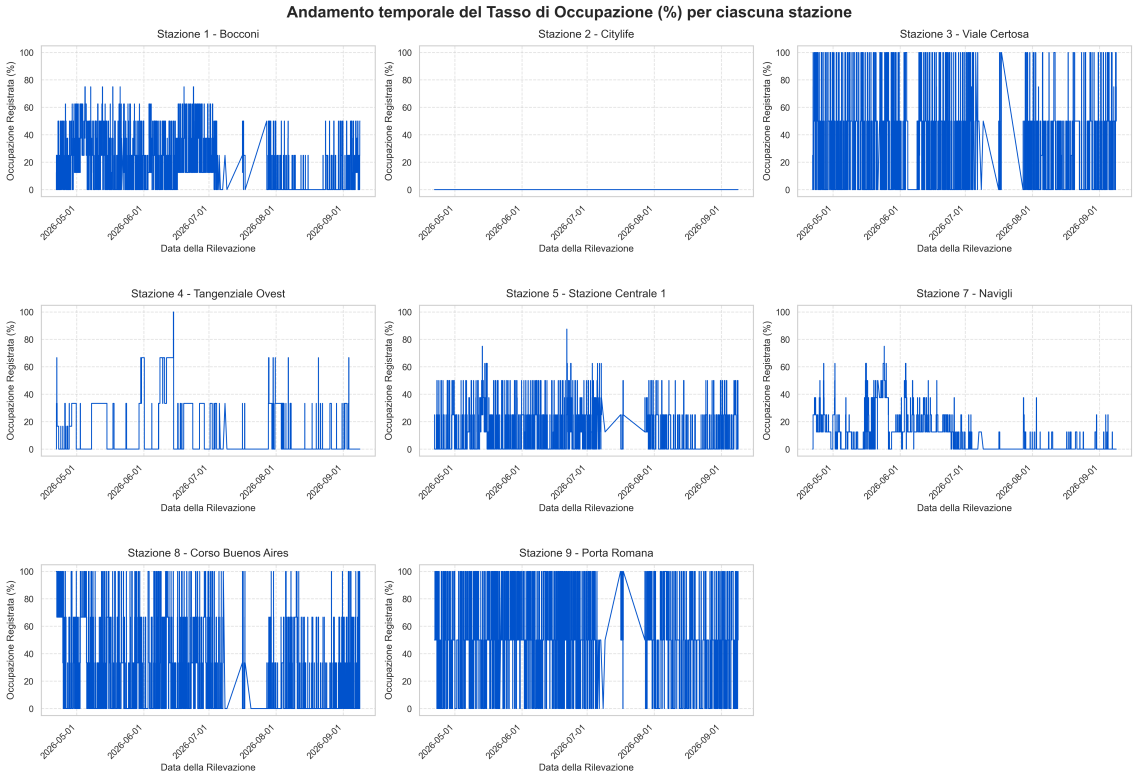


Figura 2.8: Andamento temporale dell'occupazione per singola stazione

L'ispezione visiva evidenzia una marcata discontinuità strutturale nell'orizzonte di rilevamento. Nello specifico, si osserva una prolungata assenza di dati a partire dalla sera del 6 luglio fino al pomeriggio del 27 luglio. Tale anomalia sistemica è riconducibile a una modifica restrittiva sui limiti di chiamate API giornaliere applicata dal fornitore del servizio, che ha temporaneamente inibito la capacità di interrogazione del software di raccolta. All'interno di questo esteso intervallo mancante è presente un'unica eccezione: le giornate

del 16 e 17 luglio, durante le quali il campionamento è stato transitoriamente ripristinato. Si segnala, inoltre, il caso particolare della colonnina 2 in Citylife che, pur non avendo registrato variazioni di occupazione per la quasi totalità del periodo di osservazione, è stata deliberatamente mantenuta nel dataset definitivo in quanto unica infrastruttura disponibile a copertura della relativa zona urbana. Oltre a evidenziare il periodo mancante, il grafico conferma la natura fortemente impulsiva dell'impegno dell'infrastruttura (funzione a gradino), che alterna fasi di inutilizzo a repentine saturazioni. La frammentazione temporale qui documentata ribadisce la necessità di una robusta fase di regolarizzazione del dataset prima dell'addestramento dei modelli predittivi.

Filtrando i record validi relativi a questo sottoinsieme, l'indagine si è focalizzata sulle metriche di utilizzo. La Figura 2.9 mette a confronto la prese totali per singola zona con il volume medio di prese attivamente occupate. Il tasso medio globale di occupazione si attesta al 17.18%, indicando che la rete risulta storicamente sovradimensionata rispetto alla domanda nominale media.

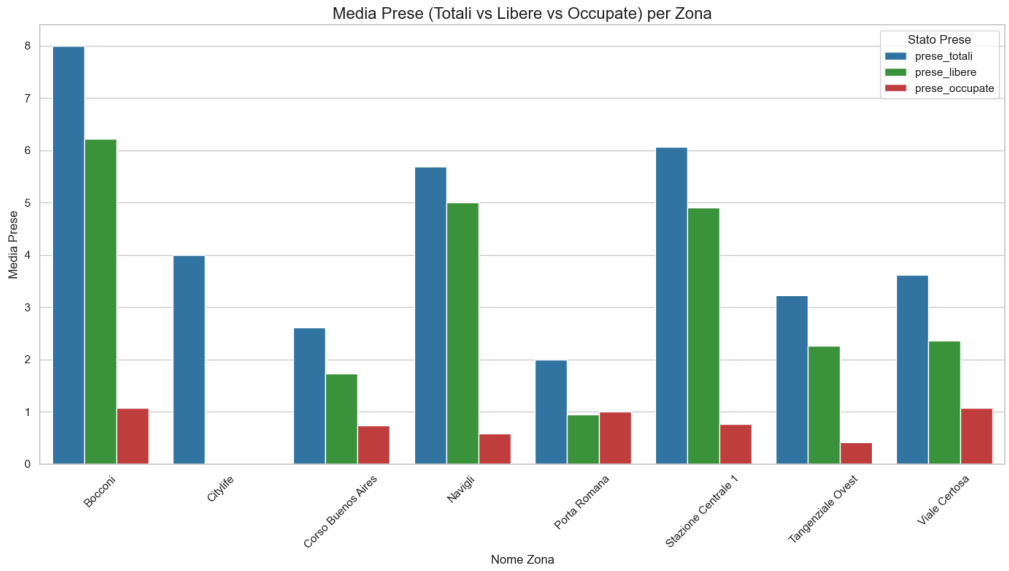


Figura 2.9: Confronto medio tra prese totali installate, prese libere e prese attivamente occupate per zona

Tuttavia, l'analisi del solo valore medio maschera l'alta variabilità del fenomeno. Il *boxplot* in Figura 2.10 descrive la dispersione statistica del tasso di occupazione. Si rileva una forte asimmetria e la presenza di *outlier*: aree come Navigli (Zona 7) registrano un utilizzo minimo, mentre zone come Porta Romana (Zona 9) e Viale Certosa (Zona 3) subiscono escursioni che portano il sistema alla massima capacità.

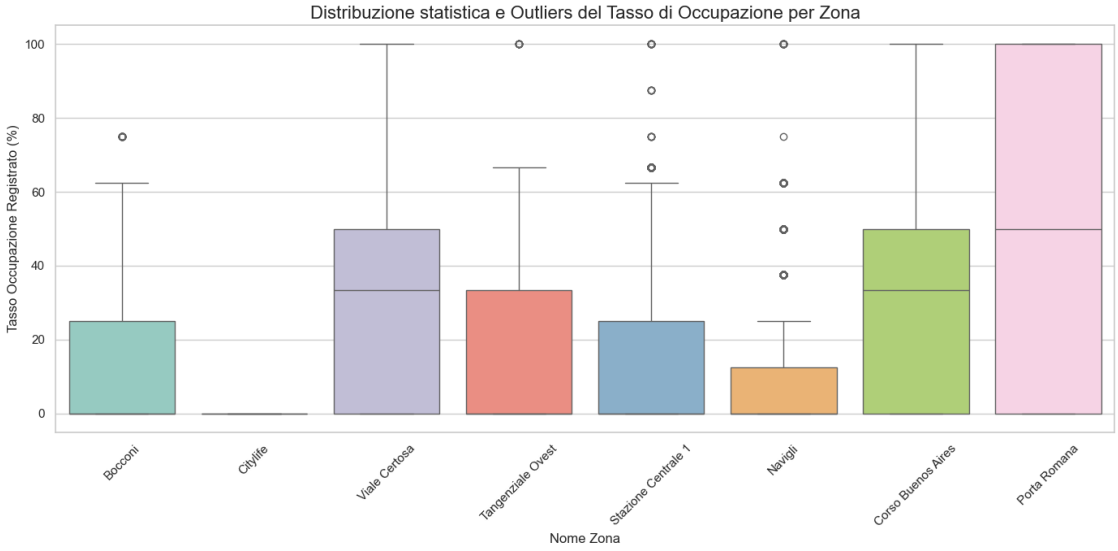


Figura 2.10: Distribuzione statistica e dispersione del tasso di occupazione per singola zona

Questa varianza determina il tasso di saturazione (Figura 2.11), ovvero la probabilità empirica che l'infrastruttura risulti completamente occupata (zero prese libere). L'analisi rivela un forte sbilanciamento territoriale: se da un lato stazioni come Porta Romana raggiungono quasi il 30% di probabilità di saturazione, dall'altro lato diverse zone (Tangenziale Ovest, Stazione Centrale 1 e Bocconi) non hanno mai registrato un singolo evento di occupazione totale nel periodo di osservazione. Va sottolineato, infine, il caso anomalo di Citylife, il cui tasso nullo non è indicativo di un mancato utilizzo, bensì di un limite hardware: ispezionando i dati grezzi, la colonnina risultava perennemente offline ("Nessun Connettore Trovato").

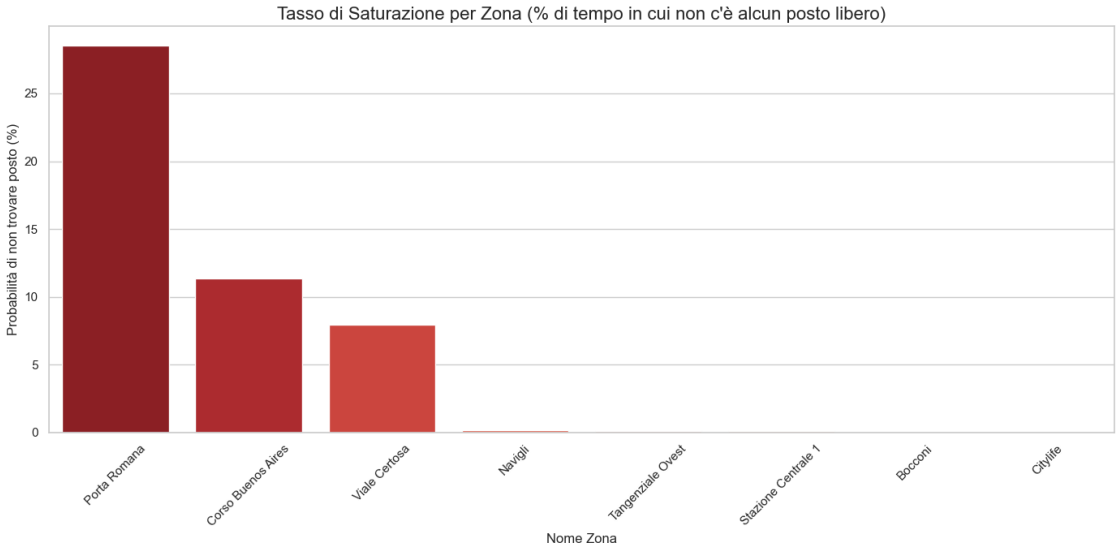


Figura 2.11: Tasso di saturazione per zona, inteso come la percentuale di tempo in cui l'infrastruttura risulta completamente occupata (zero prese libere)

L'analisi dell'evoluzione temporale della domanda mostra dinamiche strutturate. La distribuzione settimanale (Figura 2.12) evidenzia una flessione dell'occupazione durante i giorni festivi (sabato e domenica) rispetto al picco centrale registrato tra mercoledì e giovedì. Questo andamento suggerisce una dipendenza parziale della ricarica dagli spostamenti lavorativi e feriali.

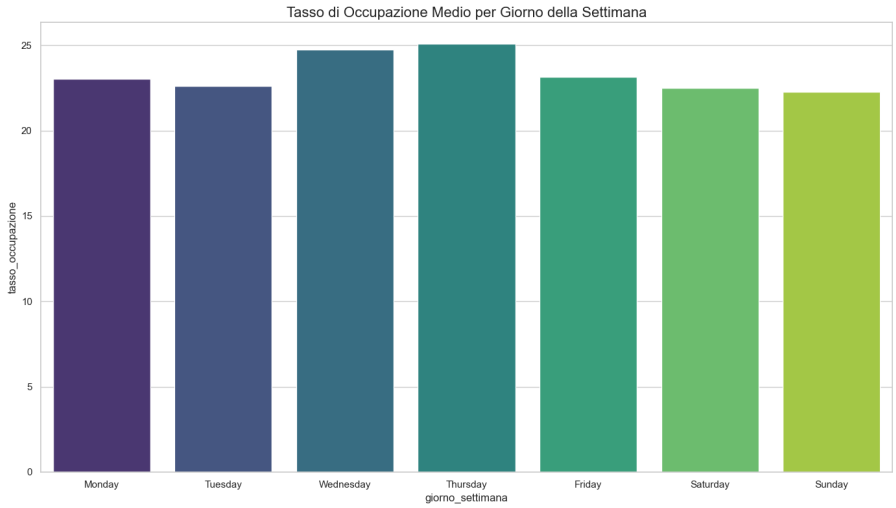


Figura 2.12: Andamento del tasso di occupazione medio aggregato lungo i giorni della settimana

Il profilo giornaliero aggregato (Figura 2.13) mostra una curva unimodale. L'occupazione decresce nelle ore notturne, inizia una fase di salita dalle 08:00, mantiene un plateau costante tra le 12:00 e le 19:00, e infine decade rapidamente nella fascia serale.

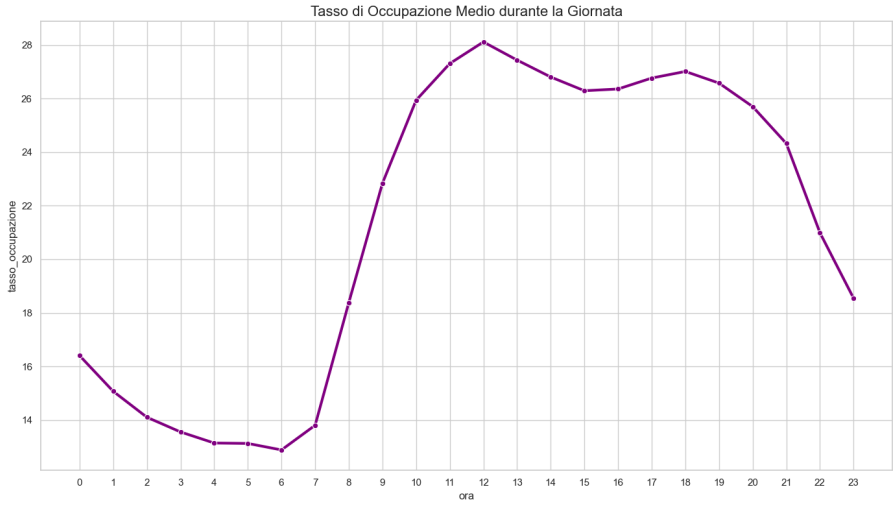


Figura 2.13: Trend orario del tasso di occupazione medio aggregato per l'intera infrastruttura di ricarica

Disaggregando il trend orario per singola zona (Figura 2.14), emergono differenze locali. Le stazioni adiacenti agli assi extraurbani mantengono profili di occupazione stabili, mentre

le colonnine situate in quartieri a forte densità evidenziano gradienti di crescita e scarico marcati.

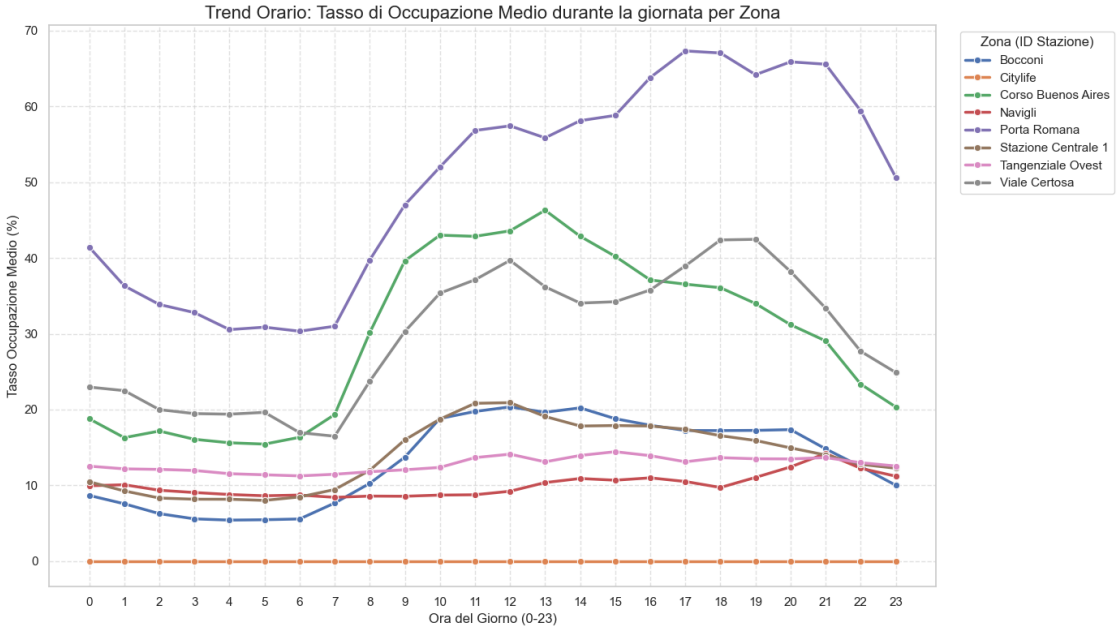


Figura 2.14: Dettaglio del trend orario di occupazione per singola zona.

A sintesi delle dinamiche spazio-temporali, la Figura 2.15 riassume il tasso di occupazione medio tramite una mappa di calore. Il confronto diretto con l'equivalente mappa del traffico stradale (Figura 2.4) definisce la problematica centrale dell'elaborato.

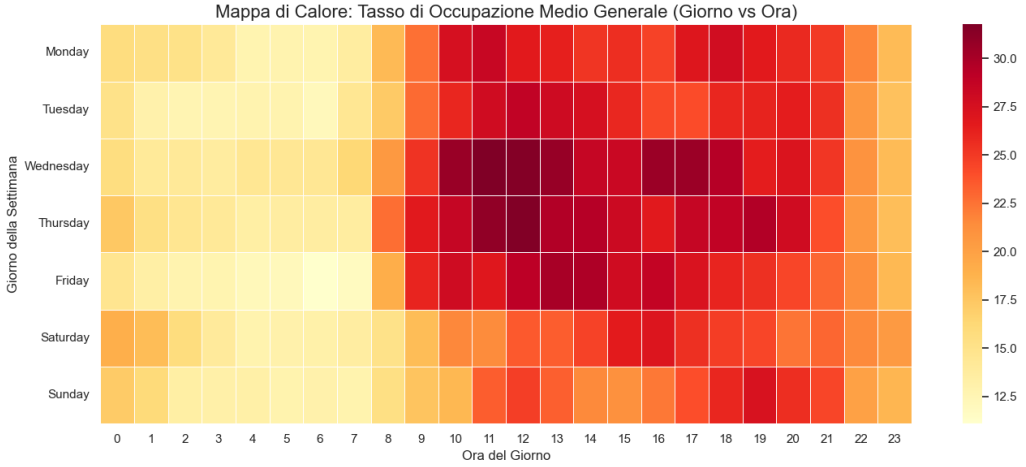


Figura 2.15: Heatmap del tasso di occupazione medio generale in funzione del giorno e dell'ora.

Mentre la congestione stradale segue fluttuazioni continue e bidirezionali (doppio picco feriale), l'occupazione di una presa EV è un evento discontinuo ad alta varianza. La connessione del veicolo genera un aumento a gradino nel segnale temporale. Il tempo di

stazionamento non è predicibile in base alla sola ora del giorno, poiché dipende dallo stato di carica della batteria e da decisioni umane. Questa asimmetria tra i due fenomeni fisici giustifica la ricerca di modelli predittivi avanzati, capaci di estrarre relazioni non lineari laddove l'autocorrelazione statistica classica risulta insufficiente.

2.4 Pre-processing e pulizia dei dati

I dati grezzi acquisiti tramite le interrogazioni API necessitano di una fase di normalizzazione prima di poter essere forniti in input agli algoritmi di apprendimento automatico. Le operazioni descritte in questa sezione mirano a rimuovere le anomalie strumentali, raccordare i formati dei dati, estrarre variabili temporali esplicite (*Feature Engineering*) e ripristinare la continuità strutturale delle serie storiche.

Data l'eterogeneità dei fenomeni osservati, le logiche di pulizia e imputazione sono state differenziate e applicate in due fasi distinte per il dataset sul traffico e per quello delle colonnine di ricarica.

2.4.1 Pre-processing del dataset sul traffico e ricostruzione dati

Il primo intervento sul dataset del traffico ha riguardato la pulizia e la formattazione formale dei record. I campi testuali originali `giorno` e `orario` sono stati fusi e convertiti in un unico oggetto `timestamp`, necessario per indicizzare temporalmente il *dataframe* [8].

Le misurazioni di velocità, lette inizialmente come stringhe, sono state forzate al formato numerico. Successivamente, è stata calcolata la metrica Δv sottraendo la `velocita_attuale` dalla `velocita_senza_traffico`. Per gestire i casi di errore di rilevamento strumentale, in cui la velocità registrata risultava superiore a quella in condizioni di scorrimento libero, è stata applicata una funzione di *clipping* che ha forzato a zero tutti i valori negativi.

Il *dataframe* è stato quindi sottoposto a un'operazione di *merge* con l'anagrafica delle stazioni, utilizzando l'identificativo della stazione come chiave di join. I record contenenti valori nulli (*NaN*) sui campi critici sono stati rimossi [8]. Dal `timestamp` ripulito sono state infine estratte le feature temporali cicliche `month` (mese), `hour` (ora del giorno) e `weekday` (giorno della settimana), che agiranno come regressori di base per i modelli di Machine Learning.

Regolarizzazione della griglia temporale Terminata la pulizia dei singoli record, l'attenzione si è spostata sull'asse temporale. Le moderne architetture predittive (in particolare i modelli SARIMA e le reti LSTM) richiedono array di input cronologicamente continui e uniformemente spaziati. I dati grezzi, tuttavia, presentavano lievi disallineamenti di campionamento (es. log alle 10:02 e il successivo alle 10:13).

Per uniformare la serie, è stata generata una griglia temporale continua con granularità bloccata a intervalli esatti di 10 minuti (es. 10:00, 10:10, 10:20). Ogni misurazione reale è stata ricondotta al punto temporale più vicino. I nodi temporali sprovvisti di misurazioni associate sono stati inizializzati come valori nulli. Questa operazione ha isolato i

confini esatti dei periodi di assenza dati senza alterare la frequenza di campionamento del fenomeno.

Progettazione dell'algoritmo di ricostruzione La fase di EDA, come visto nella figura 2.3, ha evidenziato un periodo mancante nel mese di luglio 2025. L'impiego di tecniche base, come l'interpolazione lineare, per coprire un vuoto di tali dimensioni avrebbe generato curve piatte, eliminando la varianza intrinseca del traffico e l'alternanza feriale/festiva.

È stato quindi sviluppato un algoritmo di campionamento per similarità (*Similarity-Based Sampling*) strutturato in quattro fasi:

1. **Ottimizzazione locale tramite Grid Search:** L'algoritmo adatta i propri parametri alla specifica stazione. Prima di analizzare il gap reale, isola 7 giorni di rilevazioni precedenti e 7 successive, nascondendole al modello. Attraverso una procedura di *Grid Search*, calcola il *Mean Absolute Error* (MAE) [7] per trovare la combinazione ottimale di *n_candidates* (numero di giornate storiche da valutare) e *n_sample* (numero di campioni da estrarre) su quel nodo stradale.

2. **Scoring Gerarchico Bidirezionale:** Per ogni *timestamp* nullo, l'algoritmo interroga lo storico valutando i *pattern* temporali affini. La metrica si basa su una gerarchia fissa: +100 punti per l'ora esatta (tutelando il ciclo circadiano), +50 punti per il giorno della settimana (tutelando la componente feriale/festiva) e un bonus fino a +20 punti in base alla recenza temporale. La ricerca è bidirezionale e premia i dati cronologicamente adiacenti all'interruzione.

3. **Generazione con iniezione di rumore:** Estratti i candidati, l'imputazione si basa su un campionamento casuale. Valutando la deviazione standard locale dei giorni candidati, il sistema somma al valore medio un rumore gaussiano. Questa perturbazione riproduce l'instabilità naturale del traffico e impedisce la generazione di serie eccessivamente lisce.

4. **Raccordo lineare (*Smoothing*):** Per evitare discontinuità nette nei punti di giunzione, i primi e gli ultimi campioni generati subiscono una media ponderata progressiva con i dati reali adiacenti.

Validazione statistica e visiva del dataset ricostruito L'efficacia dell'imputazione è stata validata sulle 9 stazioni stradali, incrociando l'analisi distributiva oraria, la stabilità della varianza e l'ispezione della transizione temporale pre e post-gap. Per ragioni di sintesi, vengono di seguito discussi i risultati relativi a due stazioni emblematiche: la Stazione 1, rappresentativa del traffico urbano standard, e la Stazione 5, identificata come nodo ad alta criticità.

La Figura 2.16 illustra la validazione della ricostruzione sulla Stazione 1 (Bocconi). Il pannello della continuità temporale evidenzia un raccordo privo di discontinuità ai bordi

dell'interruzione. Il profilo orario medio del mese imputato (Luglio) si sovrappone fedelmente a quello dei mesi storici adiacenti (Giugno e Settembre), mentre il grafico di distribuzione conferma che la frequenza degli eventi di traffico non ha subito alterazioni artificiali.

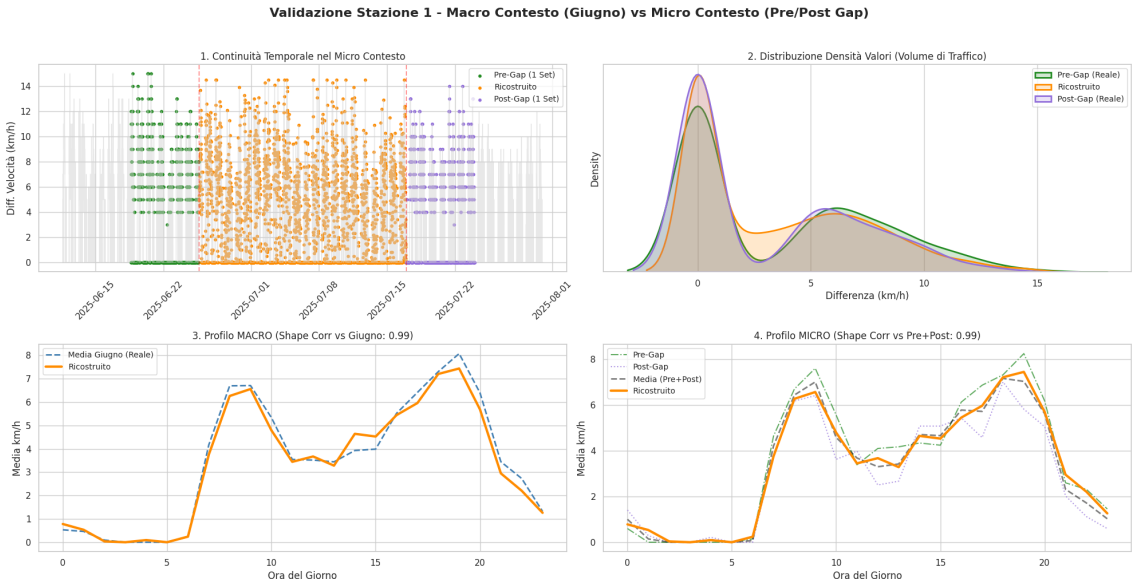


Figura 2.16: Validazione visiva e distributiva Stazione 1 (Bocconi)

Il rigore dell'imputazione sulla medesima stazione è dimostrato dall'analisi statistica (Figura 2.17). La matrice di correlazione restituisce un coefficiente molto elevato ($r \geq 0.94$) tra il mese imputato e i mesi adiacenti. I *boxplot* orari confermano che le mediane della variabile Δv e la dispersione sono perfettamente allineate ai mesi reali, certificando la riuscita dell'imputazione.

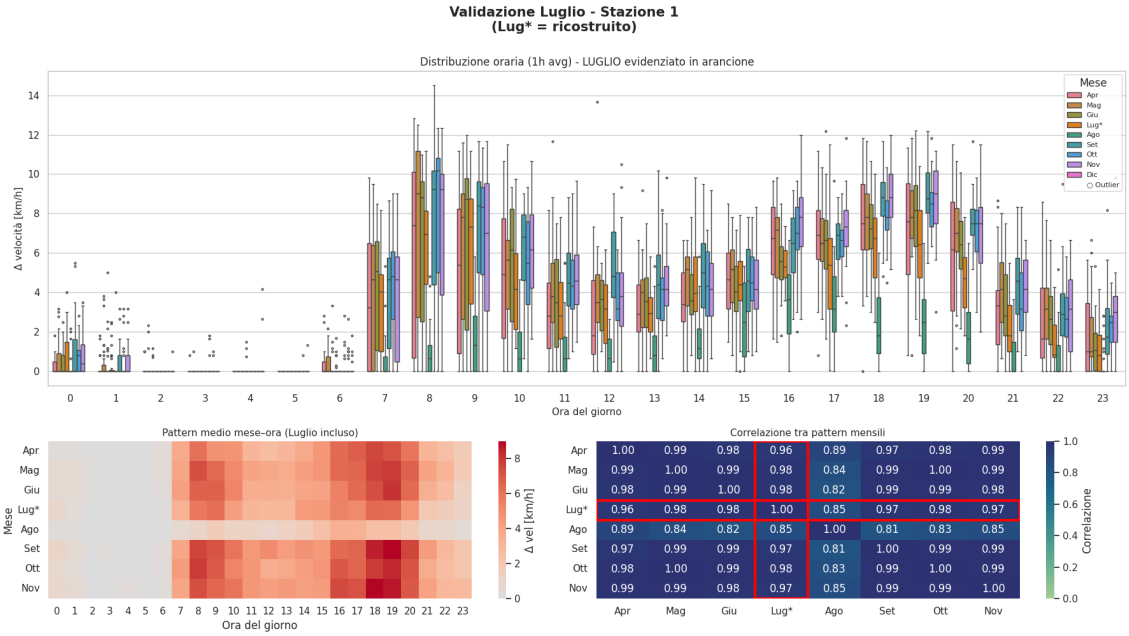


Figura 2.17: Analisi di coerenza di ricostruzione Stazione 1 (Bocconi)

La Stazione 5 (Stazione Centrale 1) presenta invece dinamiche più caotiche. La Figura 2.18 dimostra come l'algoritmo sia riuscito a preservare la morfologia della congestione: la curva di densità rispetta le "code" della distribuzione, confermando che il sistema non ha appiattito l'andamento limitandosi a prevedere il traffico scorrevole.

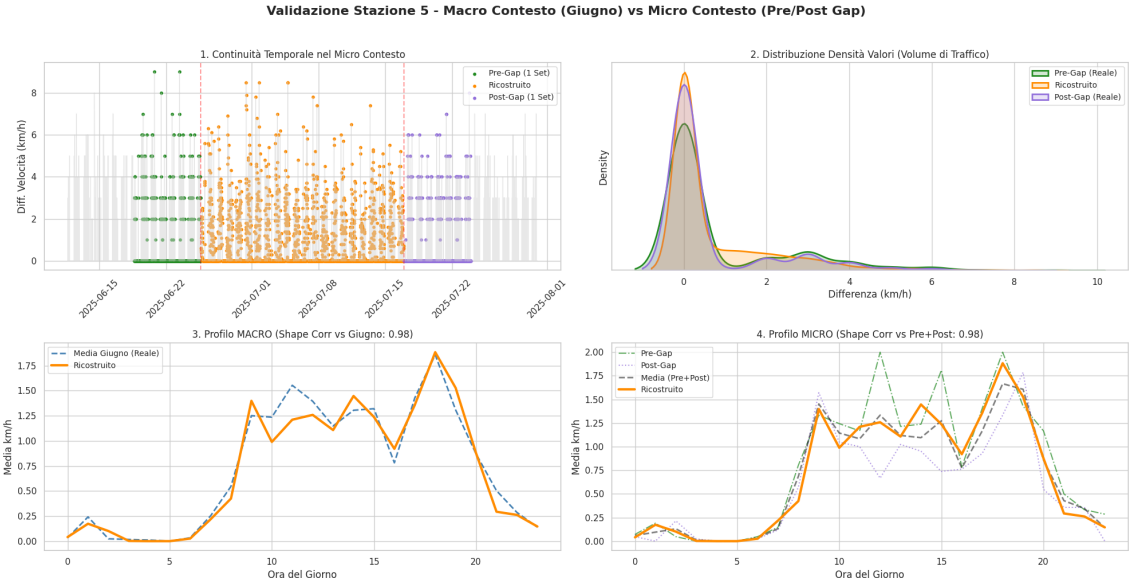


Figura 2.18: Validazione visiva e distributiva Stazione 5 (Centrale 1)

La Figura 2.19 evidenzia la bontà dell'ottimizzazione locale per la stazione 5. In questo contesto dianamico, il sistema ha ricostruito molto bene il periodo mancante, preservando con efficacia i picchi di alta congestione e garantendo al contempo valori ottimi nella matrice di correlazione ($r \approx 0.89 - 0.97$).

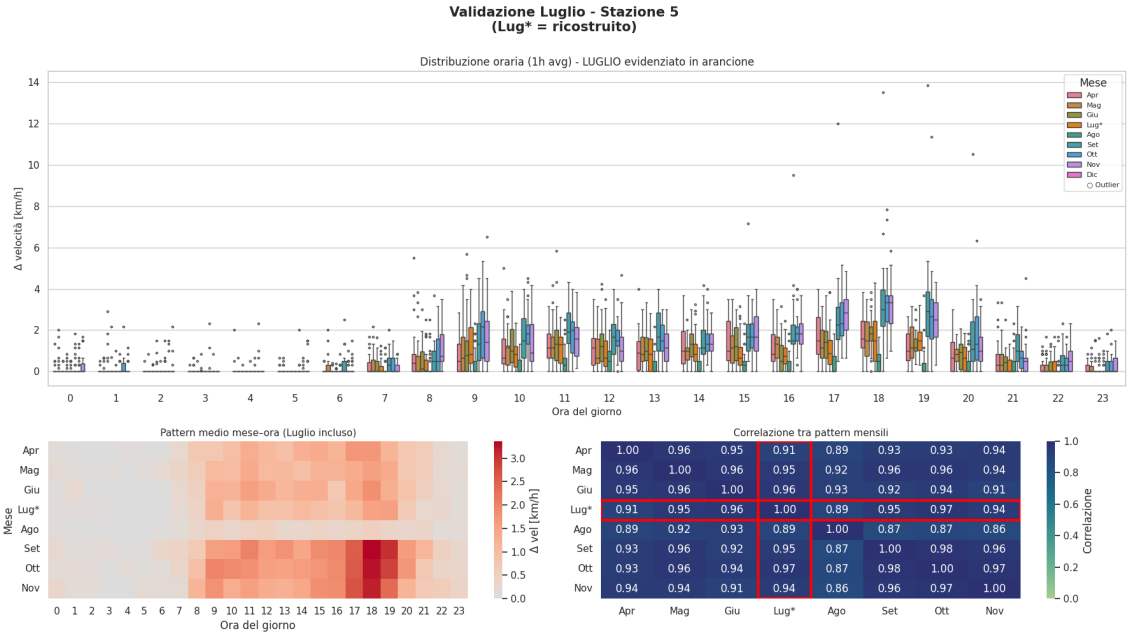


Figura 2.19: Dashboard statistico Stazione 5 (Centrale 1)

I risultati ottenuti su tutte le stazioni di rilevamento, ben rappresentati dai due casi in esame, dimostrano la capacità dell'algoritmo di ricostruzione di adattarsi a dinamiche di traffico eterogenee. Il processo elimina le discontinuità strutturali, garantendo ai modelli di Machine Learning un dataset continuo e affidabile.

2.4.2 Pre-processing del dataset EV e imputazione dati

Il trattamento delle serie storiche relative all'infrastruttura di ricarica ha richiesto un approccio metodologico specifico, dettato dalla natura discreta della variabile target (numero di prese occupate) e dalla marcata instabilità documentata nella fase esplorativa. La pipeline di elaborazione è stata strutturata in passaggi sequenziali al fine di eliminare le anomalie, allineare i riferimenti temporali e raccordare i periodi con assenza di dati.

Filtraggio spaziale e gestione degli errori di rete Il dataset grezzo è stato sottoposto a una prima scrematura spaziale, isolando in via esclusiva le registrazioni appartenenti agli 8 identificativi hardware (`id_colonnina`) con il maggiore *Score Informativo*. Le rilevazioni provenienti da stazioni di prossimità ridondanti sono state scartate per mantenere coerenza logica.

Successivamente, è stata affrontata l'anomalia presente nel dataset causata della saturazione della quota API mensile, registrata nei log con l'etichetta testuale *"tutte le chiavi esaurite"*. Poiché un blocco temporaneo di comunicazione con il server non implica una variazione fisica dello stato della colonnina, i record invalidati da tale errore sono stati convertiti in valori nulli e ripristinati tramite la propagazione dell'ultimo stato valido noto (*forward-fill*) [8], garantendo il mantenimento dell'effettivo livello di occupazione fino al ripristino della funzionalità della colonnina da parte del gestore.

Risoluzione delle anomalie di logging Un controllo di coerenza algebrica ha rivelato un difetto di tracciamento nei log antecedenti la fine di aprile 2026, finestra temporale in cui l'infrastruttura ometteva la registrazione degli stati sconosciuti. La discontinuità è stata risolta analiticamente: per ogni punto temporale in cui la somma logica tra prese libere e occupate risultava inferiore alla capacità nominale totale della singola colonnina, la differenza è stata calcolata e riassegnata alla variabile `prese_sconosciute`. I valori nulli residui relativi a stati accessori (prese fuori servizio o riservate) sono stati forzati a zero.

Allineamento temporale Per uniformare i dati ai requisiti di continuità dei modelli di Machine Learning, l'asse temporale è stato discretizzato. I *timestamp* asincroni generati dalle latenze di rete sono stati approssimati su una griglia fissa con granularità di 15 minuti. Eventuali registrazioni multiple per lo stesso ID all'interno del medesimo slot temporale sono state rimosse, conservando unicamente l'ultimo log valido per massimizzare l'attualità dell'informazione. Al fine di operare su una finestra temporale stabile e priva di anomalie, il limite cronologico inferiore del dataset è stato fissato alle ore 00:00 del 22 aprile 2026.

Risoluzione dell'interruzione temporale Il vincolo stringente sui limiti API ha generato una grave interruzione delle rilevazioni durante il mese di luglio, documentata nella precedente analisi esplorativa. L'imputazione stocastica, utilizzata con successo per il traffico veicolare, si è rivelata inadatta per il dominio EV; ricostruire matematicamente tre settimane di una grandezza impulsiva (assimilabile a una funzione a gradino) avrebbe introdotto un volume di dati sintetici superiore alla soglia di tolleranza, distorcendo l'addestramento.

Per ripristinare la continuità cronologica, è stata implementata una tecnica di *Time-Shifting*. L'algoritmo sviluppato ha isolato il segmento pre-interruzione (terminato regolarmente domenica 5 luglio 2026 alle 23:45) e il blocco di ripresa post-interruzione (avviato in modo continuo lunedì 27 luglio 2026 alle 00:00). L'intero blocco di dati post-interruzione è stato traslato all'indietro sull'asse temporale applicando un *offset* di esattamente 21 giorni (3 settimane).

In virtù di questa traslazione:

- La serie storica interrotta la domenica sera si ricongiunge senza discontinuità con le rilevazioni del lunedì mattina.
- L'alternanza feriale-festiva viene preservata in modo ineccepibile.
- La ciclicità oraria circadiana rimane inalterata, salvaguardando il *pattern* di carico di base.

Le eventuali lacune residue, generate dall'assestamento della traslazione, sono state coperte sfruttando l'autocorrelazione a 7 giorni, sovrascrivendo i dati mancanti con i valori storicizzati nel medesimo slot della settimana adiacente (passata o futura). Solo in caso di

indisponibilità di ambo i riferimenti, i *missing values* sono stati calcolati tramite interpolazione lineare discreta [7], arrotondando il risultato all'intero più vicino nel rispetto della semantica fisica (stalli interi).

La Figura 2.20 documenta l'output finale della *pipeline* di ripristino.

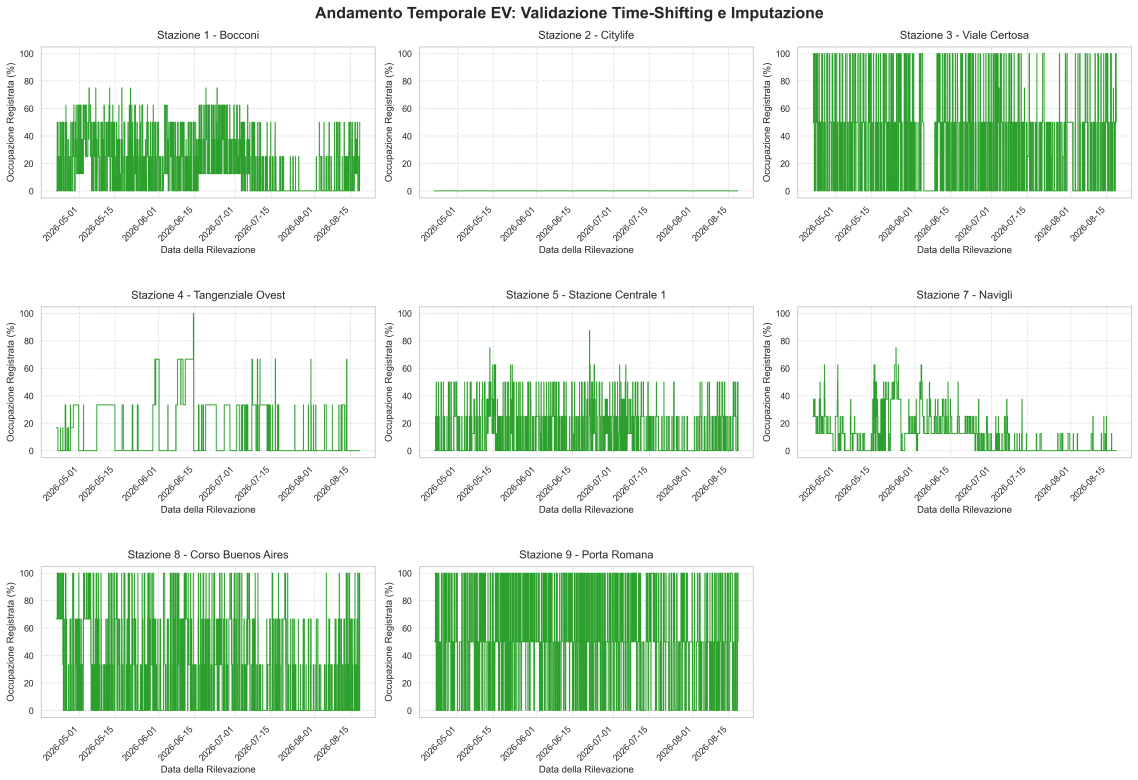


Figura 2.20: Validazione visiva post-ricostruzione

Il grafico a griglia attesta il riallineamento temporale della serie storica. Questa fase di pre-processing ha permesso di ottenere un dataset (`stato_colonnine_pulito_finale.csv`) che può essere utilizzato come input per la fase di *training*.

Capitolo 3

Metodologia e Modelli Predittivi

Questo capitolo illustra il quadro teorico e le motivazioni scientifiche alla base degli algoritmi selezionati per il lavoro di predizione. La trattazione si concentra esclusivamente sull'impianto metodologico e sul funzionamento matematico dei modelli, separando la teoria dalla successiva configurazione sperimentale.

3.1 Approccio alla previsione delle serie storiche

La modellazione e la previsione di serie storiche complesse, quali i flussi di traffico urbano e l'occupazione delle infrastrutture di ricarica, richiedono strumenti in grado di gestire simultaneamente trend di fondo, stagionalità multiple e dinamiche stocastiche. La letteratura di settore distingue diverse macro-famiglie di approcci: i metodi statistici classici, i modelli decompositivi e le tecniche di *Machine Learning* e *Deep Learning*, che privilegiano l'apprendimento dai dati attraverso approssimatori funzionali non lineari [7].

Per affrontare la natura complessa e periodica dei dati oggetto di studio, si è scelto di implementare e porre a confronto algoritmi provenienti da tutte queste famiglie. Affiancare modelli basati sull'autocorrelazione lineare ad architetture ad albero e reti neurali ricorrenti permette di valutare empiricamente quale sia più idoneo a catturare le non-linearità del dominio applicativo.

Il task di stima dell'occupazione e dei flussi di traffico è formulato matematicamente come un problema di regressione multivariata su serie temporali. Data una serie storica di osservazioni $Y = \{y_1, y_2, \ldots, y_t\}$ registrate a intervalli regolari, e un insieme di variabili esogene temporali $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t\}$ (quali ora del giorno e giorno della settimana), l'obiettivo è stimare il valore futuro y_{t+h} per un orizzonte predittivo h.

La formulazione generale della funzione di mappatura f che il modello deve apprendere è definita come:

$$\hat{y}_{t+h} = f(y_t, y_{t-1}, \ldots, y_{t-k}, \mathbf{x}_{t+h}) + \epsilon_{t+h} \tag{3.1}$$

dove k rappresenta la profondità del ritardo temporale (*lag*) considerato e ϵ_{t+h} indica il termine di errore residuo.

3.2 Algoritmi selezionati

Questa sezione descrive nel dettaglio il funzionamento teorico delle diverse famiglie di algoritmi scelti per la fase di modellazione.

3.2.1 Modelli Statistici Classici: SARIMA

Il modello SARIMA (*Seasonal AutoRegressive Integrated Moving Average*) rappresenta l'estensione del classico algoritmo ARIMA [7] per serie storiche caratterizzate da periodicità ricorrenti. La sua architettura è formalizzata dalla notazione $(p, d, q) \times (P, D, Q)_m$, dove i parametri in minuscolo definiscono la componente non stagionale e i parametri in maiuscolo la componente stagionale con periodo m.

L'equazione generale del processo stocastico è espressa come:

$$\phi_p(L)\Phi_P(L^m)\nabla^d\nabla_m^D y_t = \theta_q(L)\Theta_Q(L^m)\epsilon_t \tag{3.2}$$

dove:

- L è l'operatore di ritardo (*lag operator*), tale per cui $L^k y_t = y_{t-k}$;
- $\nabla^d = (1-L)^d$ e $\nabla_m^D = (1-L^m)^D$ sono gli operatori di differenziazione, necessari per rendere la serie stazionaria rimuovendo trend e derive stagionali;
- ϕ_p e Φ_P sono i polinomi autoregressivi (AR), che quantificano la dipendenza lineare dalle osservazioni passate;
- θ_q e Θ_Q sono i polinomi a media mobile (MA), che modellano l'impatto degli shock casuali passati;
- ϵ_t è il termine di errore, assunto come rumore bianco indipendente e identicamente distribuito.

Nonostante la solida validità teorica, l'applicazione di SARIMA a dati ad alta frequenza presenta limiti strutturali nel modellare stagionalità multiple sovrapposte (es. cicli giornalieri e settimanali congiunti), risultando inoltre computazionalmente oneroso nella fase di calibrazione dei parametri e richiedendo un notevole quantitativo di memoria RAM durante la successiva fase di addestramento. La stima dei coefficienti su dataset storici molto estesi impone infatti l'allocazione di matrici temporali di grandi dimensioni per il calcolo della massima verosimiglianza, esponendo il processo a un elevato rischio di saturazione della memoria (Out-Of-Memory) in ambienti operativi con risorse hardware limitate.

3.2.2 Modelli Additivi e Ibridi: Prophet e NeuralProphet

Per superare i limiti dei modelli ARIMA nella gestione di pattern stagionali complessi, sono state selezionate architetture basate sulla decomposizione del segnale e sulla loro recente evoluzione neurale.

Prophet

Sviluppato da Facebook [9] per serie storiche caratterizzate da forti effetti periodici, la struttura di Prophet si basa su un Modello Additivo Generalizzato (GAM):

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{3.3}$$

I regressori interni sono definiti secondo la seguente logica:

- **Trend** $g(t)$: modella le derive non periodiche. Utilizza tipicamente un modello lineare a tratti (*piecewise linear*), in cui i punti di variazione della pendenza (*changepoints*) sono selezionati automaticamente.
- **Stagionalità** $s(t)$: approssima i cicli periodici multi-frequenza tramite lo sviluppo in serie di Fourier. Per una stagionalità di periodo P, la componente è definita da:

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right) \tag{3.4}$$

  La flessibilità della curva è governata dal parametro N: un ordine elevato permette di catturare gradienti rapidi, esponendo tuttavia il sistema a potenziale *overfitting*.
- **Eventi** $h(t)$: termine che parametrizza le discontinuità attese in corrispondenza di festività.

NeuralProphet

NeuralProphet [10] rappresenta un'estensione diretta di Prophet, implementata su framework PyTorch. Mantenendo la medesima struttura additiva (trend, stagionalità e festività), questo modello sostituisce la formulazione lineare delle componenti autoregressive e delle covariate con reti neurali *Feed-Forward* (architettura AR-Net).

Questo consente di mantenere l'interpretabilità visiva delle componenti di Prophet, delegando a una rete neurale feed-forward (FNN) l'estrazione delle dipendenze complesse di breve periodo tra i *lag* temporali. L'inclusione di NeuralProphet mira a verificare se l'integrazione neurale consenta un incremento delle metriche di accuratezza rispetto alla versione classica.

3.2.3 Modelli di Machine Learning e Deep Learning: Random Forest, SVM e LSTM

L'applicazione di queste tecniche alle serie temporali richiede la trasformazione dei dati in formato tabulare attraverso il *lag embedding*, convertendo le osservazioni passate in attributi per predire il target futuro.

Random Forest Regressor

Random Forest [11] è un algoritmo di *ensemble* che basa il proprio funzionamento sull'aggregazione di molteplici alberi di decisione, costruiti parallelamente su sottocampioni del dataset generati tramite *bagging*.

Per garantire la decorrelazione tra gli stimatori, ogni albero viene addestrato su un sottoinsieme casuale di feature per la valutazione degli *split* nodali. La stima finale $\hat{y}$ si ottiene come media aritmetica delle predizioni continue $h_k(\mathbf{x})$ prodotte dai singoli K alberi:

$$\hat{y} = \frac{1}{K} \sum_{k=1}^{K} h_k(\mathbf{x}) \tag{3.5}$$

L'algoritmo è performante nel mappare relazioni fortemente non lineari nei dati di traffico, pur non possedendo capacità di estrapolazione al di fuori del range osservato durante l'addestramento.

Support Vector Regression (SVR)

L'algoritmo SVR [12] mira a individuare un iperpiano di regressione $f(\mathbf{x})$ in grado di approssimare i target reali y_i entro un limite prestabilito, pari a ϵ. L'intuizione geometrica alla base del modello è la creazione di una "banda di tolleranza" attorno alla funzione predittiva: gli errori di predizione che ricadono all'interno di questa fascia vengono considerati nulli e quindi ignorati. Al contempo, il modello cerca di mantenere l'iperpiano il più "piatto" e regolare possibile, riducendo la complessità dei pesi per minimizzare il rischio di *overfitting*.

Matematicamente, la ricerca di questo equilibrio si traduce nella minimizzazione della seguente funzione di costo (problema primale):

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{3.6}$$

dove:

- $\frac{1}{2}\|\mathbf{w}\|^2$ è il termine di regolarizzazione strutturale che penalizza le magnitudini elevate del vettore dei pesi $\mathbf{w}$, garantendo la regolarità della funzione;
- ξ_i e ξ_i^* sono le variabili di scarto (*slack variables*), introdotte per misurare l'entità dell'errore per le singole osservazioni che ricadono al di fuori della banda di tolleranza (rispettivamente per eccesso e per difetto);
- C è l'iperparametro che governa il *trade-off*: un valore elevato penalizza fortemente gli errori che eccedono il margine consentito, mentre un valore basso privilegia la regolarità della curva tollerando deviazioni maggiori.

Poiché la formulazione appena descritta definisce strutturalmente un modello lineare, per risolvere la complessa non linearità dei flussi viabilistici il modello sfrutta il cosiddet-

to *Kernel Trick*. Questa operazione matematica permette di mappare implicitamente le variabili di input in uno spazio vettoriale ad alta dimensionalità, all'interno del quale il problema di regressione diviene trattabile linearmente mantenendo inalterata l'efficienza computazionale.

Architettura LSTM

I modelli neurali ricorrenti (RNN) tradizionali presentano criticità legate alla scomparsa del gradiente (*vanishing gradient*). Le architetture LSTM (*Long Short-Term Memory*) [13] superano questo limite ingegnerizzando una cella di memoria controllata da strutture logistiche definite *gate*. La Figura 3.1 ne illustra il flusso dell'informazione.

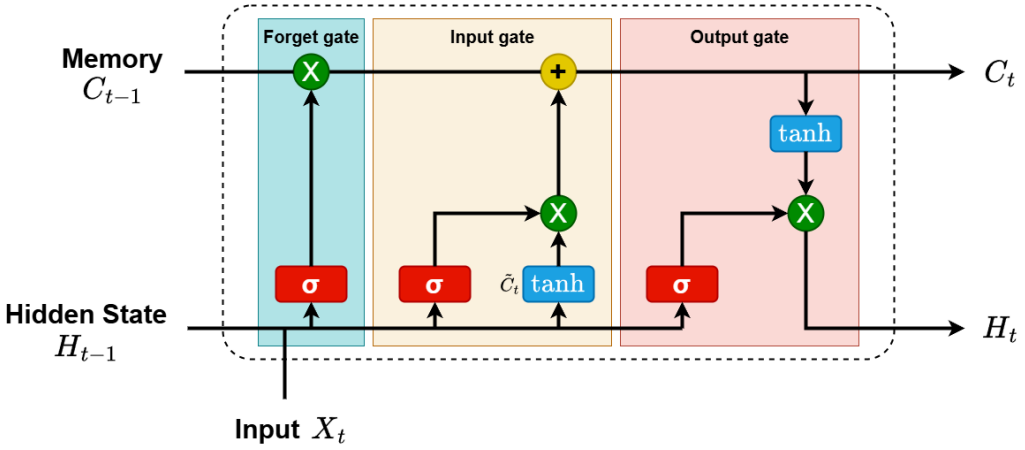


Figura 3.1: Architettura logica e flusso dell'informazione all'interno di una cella LSTM

Come illustrato nello schema, le operazioni si articolano in tre fasi principali:

- **Forget Gate ($\mathbf{f}_t$)**: riceve l'input corrente X_t e lo stato nascosto precedente H_{t-1}. Quantifica, tramite l'applicazione della sigmoide (σ), quale porzione di informazione debba essere mantenuta o rimossa dallo stato della cella C_{t-1}.
- **Input Gate ($\mathbf{i}_t$) e Candidato ($\tilde{\mathbf{C}}_t$)**: definiscono quali nuove informazioni debbano essere integrate. Il layer sigmoide funge da filtro quantitativo, mentre il layer tangente iperbolica (tanh) genera un vettore di nuovi valori candidati. La loro combinazione lineare viene sommata per determinare il nuovo stato della cella C_t.
- **Output Gate ($\mathbf{o}_t$)**: calcola l'uscita nascosta H_t filtrando il nuovo stato della cella attraverso un'ulteriore sigmoide, stabilendo così l'entità dell'informazione da propagare.

Il vettore di stato della cella C_t attraversa l'intera catena temporale agendo come una connessione lineare diretta. Questa topologia permette ai gradienti di retropropagarsi inalterati, consentendo al modello di apprendere dipendenze a lungo termine. Nell'elaborato viene adottata un'architettura *Stacked LSTM*, costituita dalla sovrapposizione di multipli strati ricorrenti per estrarre rappresentazioni gerarchiche del segnale.

3.3 Note metodologiche sui Large Time Series Models

La ricerca nell'ambito del *Time Series Forecasting* sta assistendo a un recente cambio di paradigma dovuto all'introduzione dei *Large Time Series Models* (LTSM). Architetture come *TimesFM* [14] o *Chronos* [15] utilizzano meccanismi di *Attention* e architetture *Transformer* pre-addestrate su dataset composti da miliardi di punti temporali, con l'obiettivo di eseguire predizioni in modalità *Zero-Shot*.

Nonostante le notevoli potenzialità documentate nello stato dell'arte, l'implementazione di tali modelli esula dal perimetro del presente elaborato. L'esclusione è motivata da criteri di fattibilità ingegneristica: le architetture *Transformer-based* pre-addestrate richiedono infrastrutture di calcolo (cluster GPU) e memoria RAM non compatibili con i vincoli hardware imposti per lo sviluppo locale del progetto.

Capitolo 4

Protocollo Sperimentale

Il Capitolo 4 illustra l'architettura di test e la definizione del protocollo sperimentale. L'obiettivo principale è descrivere nel dettaglio l'architettura software del test, illustrando come è stata progettata ed eseguita la fase sperimentale e motivando le scelte parametriche adottate per ciascun algoritmo. L'intera infrastruttura è stata implementata in linguaggio Python, definendo un protocollo standardizzato applicabile sia al dominio del traffico veicolare sia a quello dell'occupazione delle stazioni di ricarica.

4.1 Definizione del protocollo sperimentale

Per garantire una valutazione oggettiva degli algoritmi in condizioni operative reali, il protocollo sperimentale è stato progettato per simulare il flusso dati di un ambiente di produzione continuo. Il primo passo ha riguardato l'uniformazione della frequenza di campionamento: per garantire la compatibilità dei modelli su entrambi i domini, le serie storiche pre-processate (originariamente a 10 minuti per il traffico e 15 minuti per le colonnine) sono state ri-campionate e aggregate tramite media in slot temporali esatti da 30 minuti, generando una griglia fissa e standardizzata pari a 48 rilevazioni per ogni singola giornata.

4.1.1 Implementazione della *Sliding Window*

Il fulcro metodologico del protocollo di validazione risiede nell'adozione della tecnica della *Sliding Window* (finestra temporale scorrevole), una variante della *Cross-Validation* progettata specificamente per l'analisi delle serie storiche. Nelle classiche applicazioni di Machine Learning, il partizionamento del dataset avviene tipicamente in modo casuale e statico (es. *K-Fold Cross-Validation*); tuttavia, l'applicazione di tali metodi a dati sequenziali comporterebbe la rottura delle dipendenze temporali e introdurrebbe il grave problema del *data leakage* [16], ovvero l'utilizzo implicito di informazioni future per effettuare previsioni sul passato.

Per garantire la validità scientifica dell'esperimento e l'assenza di bias, la finestra scorrevole riproduce fedelmente il reale flusso di acquisizione temporale di un ambiente di

produzione. Il meccanismo opera secondo una logica iterativa incrementale: ad ogni passo, il modello viene addestrato sull'intero storico accumulato fino a un istante t ed è chiamato a produrre una previsione *out-of-sample* per un orizzonte futuro prefissato. Nel nostro protocollo, la finestra di previsione (`test_days`) è stata impostata rigidamente a 2 giorni (48 ore).

Terminata l'elaborazione delle metriche, la finestra di osservazione avanza lungo l'asse temporale di un passo discreto (`step_minutes`). I dati reali relativi a questo nuovo intervallo vengono resi disponibili e inglobati nel set di addestramento, permettendo al modello di riadattare i propri pesi interni per la predizione del ciclo successivo. Questo aggiornamento continuo è di fondamentale importanza nei contesti urbani per contrastare il *concept drift* [17], ovvero la fisiologica mutazione dei pattern di traffico e delle abitudini di ricarica nel tempo.

Dal punto di vista ingegneristico, il processo è stato implementato in Python avvalendosi di una funzione generatrice, riportata nel Listing 4.1. Questa specifica scelta architetturale, basata sull'istruzione `yield`, permette di ricavare le porzioni di *training* e *test* estrapolandole a *runtime* dal dataframe principale iterazione dopo iterazione. Ciò abbatte drasticamente il consumo di memoria RAM rispetto alla pre-allocazione statica di migliaia di subset.

```
from datetime import timedelta

def sliding_window_generator(df, train_days='auto', test_days=2,
    step_minutes=1440, max_folds=30, test_on_end=True):
    start_date = df.index.min()
    end_date = df.index.max()
    fold_counter = 1

    # Allineamento dell'avvio per eseguire il test a ritroso dalla fine del
     dataset
    if test_on_end:
        minuti_scorrimento = (max_folds - 1) * step_minutes
        inizio_primo_test = end_date - timedelta(minutes=minuti_scorrimento
    ) - timedelta(days=test_days)

        if train_days == 'auto':
            train_days = (inizio_primo_test - start_date).days

        current_start = inizio_primo_test - timedelta(days=train_days)
    else:
        current_start = start_date

    # Generazione dinamica dei fold temporali
    while True:
        if max_folds is not None and fold_counter > max_folds: break

        train_end = current_start + timedelta(days=train_days)
        test_end = train_end + timedelta(days=test_days)
```

```
26
27          if test_end > end_date: break
28
29          # Estrazione delle porzioni storiche e future senza data leakage
30          train_data = df[(df.index >= current_start) & (df.index < train_end
    )]
31          test_data = df[(df.index >= train_end) & (df.index < test_end)]
32
33          yield fold_counter, train_data, test_data
34
35          # Avanzamento della finestra temporale
36          current_start += timedelta(minutes=step_minutes)
37          fold_counter += 1
```

Listing 4.1: Implementazione della funzione generatrice per la Sliding Window temporale dinamica

4.1.2 Isolamento del periodo di test: Scorrimento Giornaliero e Scorrimento a 30 minuti

La corretta partizione del dataset in sottoinsiemi di *training* e *test* rappresenta un punto fondamentale. Per massimizzare la capacità di generalizzazione degli algoritmi, l'architettura di partizionamento prevede che l'intero storico ininterrotto dei dati acquisiti venga destinato all'addestramento iniziale (`train_days='auto'`).

Per esplorare in modo analitico il *trade-off* tra reattività del modello ed efficienza computazionale, il protocollo di test è stato strutturato su due scenari sperimentali comparativi:

1. **Scorrimento giornaliero (Step 24 ore):** In questa prima configurazione (`step_minutes=1440`), il modello fornisce previsioni per le 48 ore successive, ma i suoi pesi interni vengono aggiornati una sola volta al giorno. Tale approccio simula un tipico ambiente di produzione aziendale basato su elaborazioni *batch* notturne, abbattendo drasticamente il carico computazionale. In questo scenario, l'analisi sul traffico ha coperto un orizzonte di 30 giorni consecutivi (30 iterazioni totali), mentre l'analisi sull'occupazione delle stazioni di ricarica è stata valutata su una finestra di 7 giorni (7 iterazioni), proporzionata alla minore profondità storica dei dati EV a disposizione (acquisizioni da aprile a settembre 2026).

2. **Scorrimento ad alta frequenza (Step 30 minuti):** In questa seconda configurazione (`step_minutes=30`), il modello viene riaddestrato non appena si rende disponibile un nuovo campionamento. Questo approccio simula un sistema ITS (Intelligent Transportation System) [18] puramente *real-time*, capace di reagire tempestivamente alle micro-variazioni del contesto urbano. Per mantenere la coerenza metodologica, le tempistiche di test sono state allineate allo scenario giornaliero: l'esperimento sul traffico stradale è stato eseguito ininterrottamente per l'intero mese,

richiedendo giorni di calcolo e fungendo da vero e proprio *stress test* per l'architettura. Coerentemente, il test ad alta frequenza sulle colonnine EV è stato circoscritto alla singola settimana operativa (336 iterazioni).

L'implementazione dello scenario ad alta frequenza introduce infatti una severa sfida computazionale. Lavorando su misurazioni con granularità di 30 minuti, l'avanzamento continuo della finestra di valutazione (mantenendo un orizzonte predittivo costante di 48 ore) lungo un periodo complessivo di 30 giorni impone la generazione di esattamente 1440 iterazioni distinte per ogni singola combinazione di modello e stazione. Estendendo questo processo a tutte le architetture algoritmiche e ai numerosi nodi monitorati, si genera un carico elaborativo massivo, spesso considerato impraticabile nei convenzionali ambienti di produzione.

L'esecuzione parallela di questi due protocolli rappresenta una scelta strategica fondamentale per l'elaborato: permette di confrontare i risultati e stabilire oggettivamente se l'enorme dispendio di risorse richiesto da un aggiornamento continuo sia concretamente giustificato da un salto prestazionale significativo, oppure se l'approccio di ricalcolo giornaliero offra una soluzione ingegneristica più scalabile e altrettanto robusta.

4.2 Impostazione dei parametri per ciascun modello

Per garantire che il confronto delle performance predittive avvenisse ad armi pari e in modo oggettivo, i sei algoritmi selezionati (Random Forest, Neural Prophet, Stacked LSTM, Prophet, SVM e SARIMA) sono stati incapsulati in apposite classi *Wrapper* in Python [19]. Questo paradigma della programmazione a oggetti ha permesso di esporre un'interfaccia di addestramento comune standardizzando le fasi di pre-processing interno. Tra queste, riveste un ruolo chiave l'impiego del `MinMaxScaler` per le reti LSTM, una funzione che comprime linearmente i valori in un intervallo fisso (tipicamente $[0, 1]$) per garantire stabilità numerica e una convergenza più rapida dei gradienti; parimenti, per la SVM è stato applicato uno `StandardScaler`, che standardizza le *feature* rimuovendo la media e ridimensionando la varianza a 1, evitando così che variabili su scale numeriche più ampie finiscano per dominare inavvertitamente la funzione di costo geometrica dell'algoritmo. Inoltre, i *Wrapper* applicano sistematicamente a valle una funzione di *clipping* (es. `np.clip`) per forzare l'output predittivo entro limiti fisicamente sensati (ad esempio bloccando l'occupazione EV strettamente nel *range* $[0, 100]\%$).

Nel dettaglio, le configurazioni parametriche specifiche adottate per calibrare i singoli algoritmi, al fine di massimizzarne le capacità di estrapolazione e prevenire fenomeni di *overfitting*, sono le seguenti:

- **Random Forest Regressor:** Configurato con 100 alberi decisionali (`n_estimators=100`), un valore empirico ritenuto ottimale per bilanciare la stabilizzazione delle previsioni (riduzione della varianza dell'*ensemble*) e il tempo di calcolo. La profondità massima è stata limitata a 10 (`max_depth=10`) per indurre una forte regolarizzazione,

impedendo agli alberi di sviluppare rami iper-specifici che memorizzerebbero semplicemente il rumore dei dati. Poiché questo modello non possiede una memoria temporale intrinseca, è stato alimentato con *feature* cicliche esplicite (ora, giorno della settimana) e variabili di *lag* (osservazioni ritardate a 30 minuti, 1 ora e 24 ore prima) per fornirgli forzatamente il contesto autoregressivo.

- **Support Vector Machine (SVM):** Utilizza un kernel radiale non lineare (`rbf`) per mappare e decodificare le complesse fluttuazioni non lineari presenti nei pattern urbani. Il parametro di regolarizzazione è stato fissato a $C = 1.0$ per ottenere un compromesso bilanciato tra la minimizzazione dell'errore di addestramento e la regolarità della superficie decisionale. Il margine di tolleranza $\epsilon = 0.1$ impone al modello di ignorare gli errori residui inferiori a tale soglia, rendendo la retta di regressione intrinsecamente insensibile alle fisiologiche micro-oscillazioni e ai disturbi isolati dei sensori.

- **Stacked LSTM:** L'architettura profonda si basa su una finestra di *look-back* di 48 slot, progettata appositamente affinché la memoria a breve termine della rete copra l'esatto ciclo delle 24 ore precedenti (essendo le misurazioni a 30 minuti). La rete è composta da layer sequenziali di dimensioni decrescenti (64, 32 e 16 neuroni) per condensare l'informazione estraendo *feature* via via più astratte. I livelli sono intervallati da strati di `Dropout` al 20%, una funzione che azzera casualmente una frazione dei pesi neurali durante il *training*: ciò costringe la rete a sviluppare rappresentazioni multiple e ridondanti del traffico, impedendole di dipendere da singoli nodi e mitigando drasticamente l'*overfitting*. L'addestramento avviene per 20 epoche tramite ottimizzatore *Adam* (che regola automaticamente i passi di discesa del gradiente in base alla varianza storica) in lotti da 64 campioni.

- **Prophet e NeuralProphet:** Entrambi i modelli sono stati configurati forzando la disattivazione della stagionalità annuale. Avendo a disposizione un volume storico inferiore ai dodici mesi (es. da aprile a settembre 2026 per i dati EV), l'attivazione di questa componente avrebbe inevitabilmente indotto il modello a forzare curve stagionali incomplete, generando gravi distorsioni predittive. Al contrario, è stato attivato l'apprendimento esplicito della stagionalità giornaliera e settimanale per isolare sistematicamente l'alternanza giorno/notte e le differenze tra giorni feriali e festivi. Per la variante neurale, il ciclo di addestramento è stato fissato a 50 epoche.

- **SARIMA:** I parametri di derivazione e stagionalità sono stati ottimizzati empiricamente; per i nodi urbani è stata spesso impiegata la configurazione `order=(3,0,1)` e `seasonal_order=(0,0,1,48)`. In questa metrica, il blocco autoregressivo valuta gli ultimi 3 slot (1.5 ore di dipendenza breve), mentre la componente stagionale (48) mappa rigorosamente il medesimo orario del giorno precedente. Un accorgimento ingegneristico fondamentale è consistito nel forzare la differenziazione stagionale a zero ($D = 0$): tale scelta si è resa necessaria per mantenere stabilità numerica. Applicare

la differenziazione standard su serie storiche con un'alta presenza di zeri prolungati (come le stazioni sul traffico completamente assente durante le ore notturne) causerebbe infatti collassi matriciali e la generazione persistente di previsioni non valide (valori `NaN`).

A fini riepilogativi e per una rapida consultazione, la Tabella 4.1 sintetizza gli iperparametri appena descritti per ciascun algoritmo predittivo.

Modello	**Configurazione Parametrica e Iperparametri**
Random Forest	`n_estimators`=100, `max_depth`=10, `n_jobs`=-1. Feature esogene: *hour, dayofweek, is_weekend, lag_1, lag_2, lag_48*.
Support Vector Machine (SVM)	Kernel radiale (`rbf`), `C`=1.0, tolleranza all'errore $\epsilon = 0.1$. Feature esogene standardizzate tramite `StandardScaler`.
Stacked LSTM	Finestra di *look-back*=48 slot. Strati nascosti: 64, 32 e 16 neuroni. `Dropout`=0.2, `epochs`=20, `batch_size`=64, ottimizzatore *Adam*.
Neural Prophet	`epochs`=50. Stagionalità: giornaliera e settimanale attivate esplicitamente, componente annuale disattivata.
Prophet	Stagionalità: giornaliera e settimanale attivate, componente annuale disattivata. Ottimizzazione di default tramite L-BFGS.
SARIMA	`order`=(3,0,1), `seasonal_order`=(0,0,1,48). Differenziazione stagionale (D) forzata a 0 per prevenire instabilità numerica e valori NaN su serie inflazionate da zeri.

Tabella 4.1: Riassunto delle configurazioni e degli iperparametri adottati per ciascun algoritmo predittivo.

4.3 Metriche di valutazione degli errori

La valutazione quantitativa e il confronto delle architetture predittive sono stati delegati a una funzione dedicata (`run_evaluation_loop_modulare`). All'interno di questo modulo, a ogni step temporale, i vettori delle previsioni generati dai modelli vengono confrontati matematicamente con i dati reali di *ground-truth* estrapolati dal test set. Questa fase include anche tecniche per filtrare e sanificare eventuali instabilità numeriche (valori NaN o infiniti) prodotte dai modelli prima del calcolo dell'errore.

Per fornire una misurazione completa delle performance, le discrepanze tra i valori reali e quelli stimati sono state archiviate in formato tabellare e quantificate tramite tre indicatori statistici standard [7]. Denotando con N il numero totale delle osservazioni

nell'orizzonte di test, con y_i il valore reale misurato all'istante i e con $\hat{y}_i$ il valore predetto dal modello per il medesimo istante, le metriche sono definite come segue:

- **MAE (Mean Absolute Error):** Calcolato come la media aritmetica degli scarti assoluti tra le predizioni e le osservazioni reali:

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

  Questa metrica restituisce un errore interpretabile direttamente nella stessa scala di misura del fenomeno osservato (in km/h per il traffico o in percentuale puntuale di prese occupate per le colonnine EV). Essendo un indicatore lineare, attribuisce un peso proporzionale a tutti gli errori, fornendo una stima robusta e non distorta dalla presenza di eventuali outlier isolati.

- **RMSE (Root Mean Square Error):** Rappresenta la radice quadrata della media degli errori al quadrato:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}$$

  A differenza del MAE, l'operazione di elevamento al quadrato penalizza in modo severo e non lineare gli errori di magnitudo superiore. Nel contesto della viabilità e della ricarica EV, l'utilizzo combinato del RMSE si rivela cruciale: un valore di RMSE significativamente superiore al MAE indica che il modello tende a commettere errori di elevata entità in corrispondenza dei picchi di traffico imprevisti o delle saturazioni improvvise dell'infrastruttura, ovvero le casistiche di maggior disagio per l'utente finale.

- **MAPE (Mean Absolute Percentage Error):** Fornisce una misura dell'accuratezza espressa in termini percentuali relativi:

$$MAPE = \frac{100}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

  Questa grandezza permette di confrontare trasversalmente l'efficacia dei modelli su stazioni diverse, astraendo dal volume assoluto (es. nodi ad alto scorrimento, vie secondarie). A livello software, per preservare la stabilità di questa formula, la metrica è stata ingegnerizzata implementando una logica condizionale vettoriale che neutralizza matematicamente le divisioni per zero intrinseche nel calcolo del MAPE quando le stazioni risultano totalmente vuote ($y_i = 0$).

4.3.1 Analisi dell'andamento e del decadimento dell'errore

Al di là della validazione statica basata sulle medie globali, il framework di test è stato progettato per indagare l'evoluzione dinamica dell'errore predittivo lungo due dimensioni temporali distinte.

La prima dimensione di analisi riguarda il **decadimento dell'affidabilità sull'orizzonte predittivo**. Calcolare un errore medio su un blocco di 48 ore future restituisce infatti un'indicazione incompleta, poiché l'accuratezza di un modello tende fisiologicamente a peggiorare man mano che la stima si allontana dall'istante presente. Per quantificare questo decadimento, la funzione di valutazione calcola e archivia il MAE e l'errore percentuale per ogni singolo slot da 30 minuti proiettato nel futuro (`errore_assoluto_slot_i`). L'aggregazione di queste misurazioni permette di tracciare la curva di degrado dell'affidabilità del modello da 0 a 48 ore, identificando le soglie oltre le quali la stima perde la propria utilità applicativa.

La seconda dimensione si focalizza sulle **difficoltà di generalizzazione nei diversi momenti dell'anno**. A causa della marcata variabilità dei dati, i modelli sperimentano fluttuazioni di *performance* a seconda del giorno testato. Attraverso funzioni di aggregazione specifiche (`analizza_momenti_dell_anno`), le metriche di errore globali vengono raggruppate per data esatta del test e aggregate su base mensile. Questa visualizzazione storica permette di isolare i giorni peggiori per l'algoritmo predittivo e di identificare quali mesi presentino pattern di ricarica e di transito più difficili da generalizzare.

4.4 Tracciamento dei costi computazionali e latenza

La fase sperimentale, interamente automatizzata e parallelizzata dove l'hardware lo consentiva (come nel caso del Random Forest con `n_jobs=-1`), è stata profilata al fine di valutarne la fattibilità applicativa.

Oltre all'accuratezza predittiva, un requisito fondamentale per un sistema intelligente urbano (ITS) è la responsività in produzione. Per questo motivo, all'interno del loop di valutazione, la libreria nativa `time` di Python (tramite `time.perf_counter()`) è stata utilizzata per tracciare rigorosamente i costi computazionali, separandoli in due metriche:

1. **Tempo di Training (Offline):** I secondi impiegati dal modello per ricalcolare i propri pesi interni sull'intero dataset storico aggiornato.

2. **Latenza di Inferenza (Online):** I millisecondi necessari al modello pre-addestrato per restituire la previsione sul singolo slot futuro. Questo parametro certifica la scalabilità del modello e la sua idoneità ad essere interrogato in tempo reale da un cruscotto utente senza generare colli di bottiglia architetturali.

4.4.1 Ambiente software e infrastruttura di calcolo

L'intera pipeline sperimentale, dall'acquisizione dei dati fino all'addestramento dei modelli e alla generazione automatizzata dei report grafici, è stata sviluppata in Python. Per far fronte all'elevato onere elaborativo richiesto dall'iterazione della *Sliding Window* – specialmente durante il riaddestramento continuo delle reti neurali e dei modelli statistici multivariati – l'esecuzione dei codici è stata centralizzata su un server dipartimentale dedicato.

È opportuno precisare una limitazione architetturale emersa in fase di definizione degli scenari di test. Per lo scenario di scorrimento ogni 30 minuti, gli algoritmi SARIMA e Stacked LSTM sono stati esclusi dal set di valutazione. La scelta è stata dettata dai vincoli hardware dell'ambiente di calcolo: l'ottimizzazione del modello SARIMA portava il sistema a saturazione di memoria (*Out-Of-Memory*) durante il calcolo della verosimiglianza per le componenti stagionali, mentre l'assenza di una GPU rendeva i tempi di riaddestramento iterativo della rete LSTM incompatibili con la fattibilità dell'esperimento. Tali modelli sono stati regolarmente valutati in tutti gli altri scenari di test (scorrimento giornaliero), dove il volume di calcolo rientrava nei limiti dell'infrastruttura.

La Tabella 4.2 riassume le principali librerie software impiegate e i dettagli dell'ambiente operativo, garantendo la piena riproducibilità metodologica e tecnica dell'esperimento.

Componente	**Specifica e Librerie Utilizzate**
Linguaggio di programmazione	Python 3.10+
Manipolazione Dati	`pandas`, `numpy`
Machine Learning	`scikit-learn` (RandomForest, SVR, StandardScaler, MinMaxScaler, Metriche di errore)
Deep Learning	`tensorflow` / `keras` (Architettura Stacked LSTM)
Modelli Statistici Additivi	`statsmodels` (SARIMA), `prophet`, `neuralprophet` (con backend `pytorch`)
Data Visualization	`matplotlib`, `seaborn`
Infrastruttura di Calcolo	Server Dipartimentale

Tabella 4.2: Ambiente software e infrastruttura di calcolo utilizzati

Capitolo 5

Risultati e Discussione

Questo capitolo espone e analizza i risultati sperimentali ottenuti dall'applicazione della pipeline predittiva. L'esposizione è strutturata in tre fasi logiche consequenziali: dapprima si valuta l'efficacia dei modelli sul dominio del traffico stradale; successivamente, si traslano le medesime architetture sul dominio delle infrastrutture di ricarica EV; infine, si analizza la correlazione statistica tra i due fenomeni per validare l'impiego del traffico stradale come predittivo, concludendo con una discussione critica sui costi computazionali e sui *trade-off* ingegneristici emersi.

5.1 Risultati sul dataset del traffico stradale

In questa prima fase della valutazione, focalizzata sul dominio viabilistico, **la variabile target oggetto di previsione è la differenza di velocità (Δv)**, un parametro quantitativo scelto per misurare in modo oggettivo il livello di congestione e lo scostamento del flusso veicolare rispetto alle condizioni ottimali di traffico assente.

L'addestramento dei modelli sui dati storici provenienti da stazioni strategiche della città di Milano risponde a una precisa necessità metodologica. Testare l'infrastruttura algoritmica su una serie temporale continua e autocorrelata, come quella del traffico, permette infatti di validare la solidità e l'efficienza degli algoritmi in un dominio stabile. Stabilire questa baseline prestazionale rappresenta il passaggio essenziale prima di affrontare il dominio target delle colonnine EV.

5.1.1 Analisi visiva: sovrapposizione curve reali e previsioni

Il primo passo di validazione è di natura qualitativa e si basa sull'ispezione grafica della corrispondenza tra la serie storica reale e l'output predittivo generato all'interno della *Sliding Window*. L'allineamento tra le rilevazioni del traffico e le stime dei modelli è mostrato in Figura 5.1. I grafici, organizzati in riquadri per le 9 stazioni monitorate, illustrano la complessa dinamica veicolare: la linea spessa scura rappresenta il ground truth, mentre le linee sottili colorate tracciano l'output di ogni singolo algoritmo nel periodo di test.

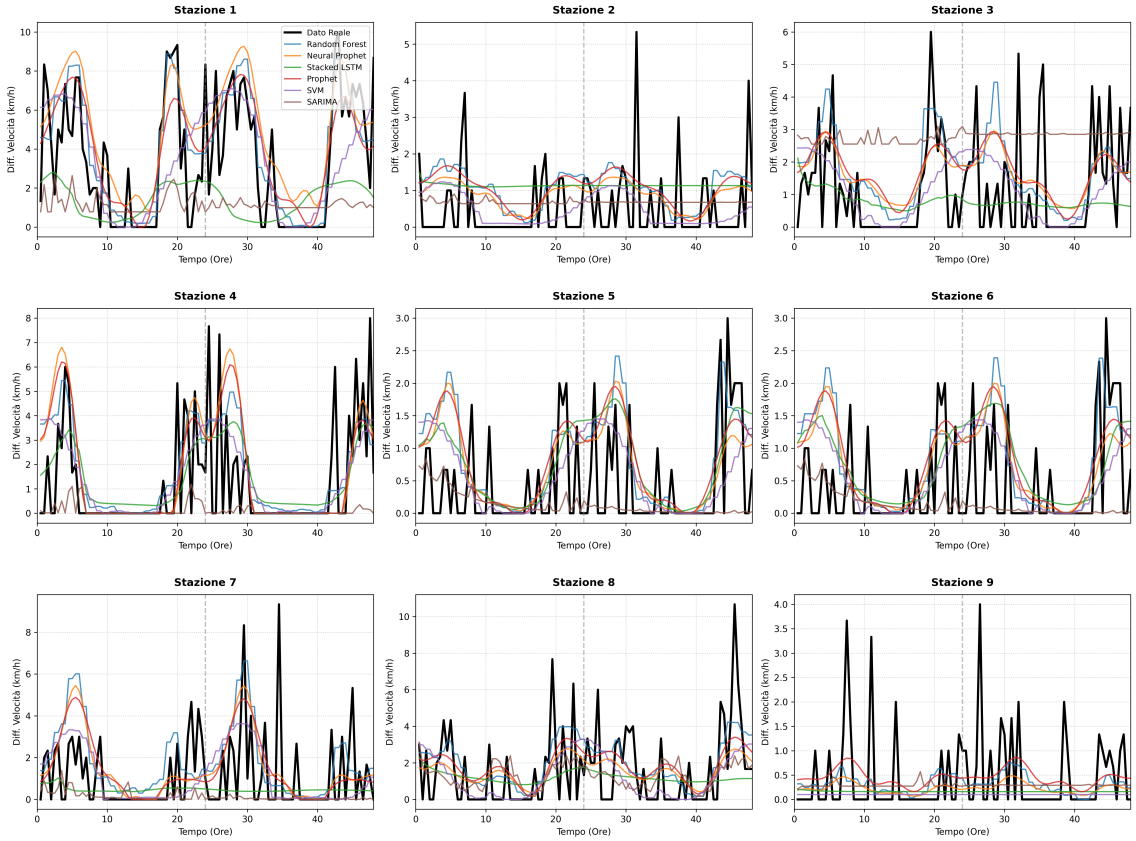


Figura 5.1: Sovrapposizione delle previsioni dei modelli sul dato reale (Δv) per le singole stazioni di monitoraggio (Scorrimento a 1 giorno)

Dall'osservazione dei grafici temporali per lo scorrimento giornaliero (Figura 5.1), l'intera architettura predittiva dimostra di riuscire a predire l'andamento generale del traffico cittadino. I modelli riproducono la ciclicità feriale del fenomeno, intercettando i picchi della congestione mattutina (tra le 07:00 e le 10:00) e i rallentamenti della fascia pomeridiana-serale (tra le 17:00 e le 20:00), in perfetto allineamento con i pattern emersi nell'Analisi Esplorativa (Capitolo 2). Spicca, inoltre, la capacità degli algoritmi di assecondare il netto calo della metrica durante le ore notturne.

La Figura 5.2 presenta l'equivalente analisi visiva per lo scenario di scorrimento ad alta frequenza (30 minuti).

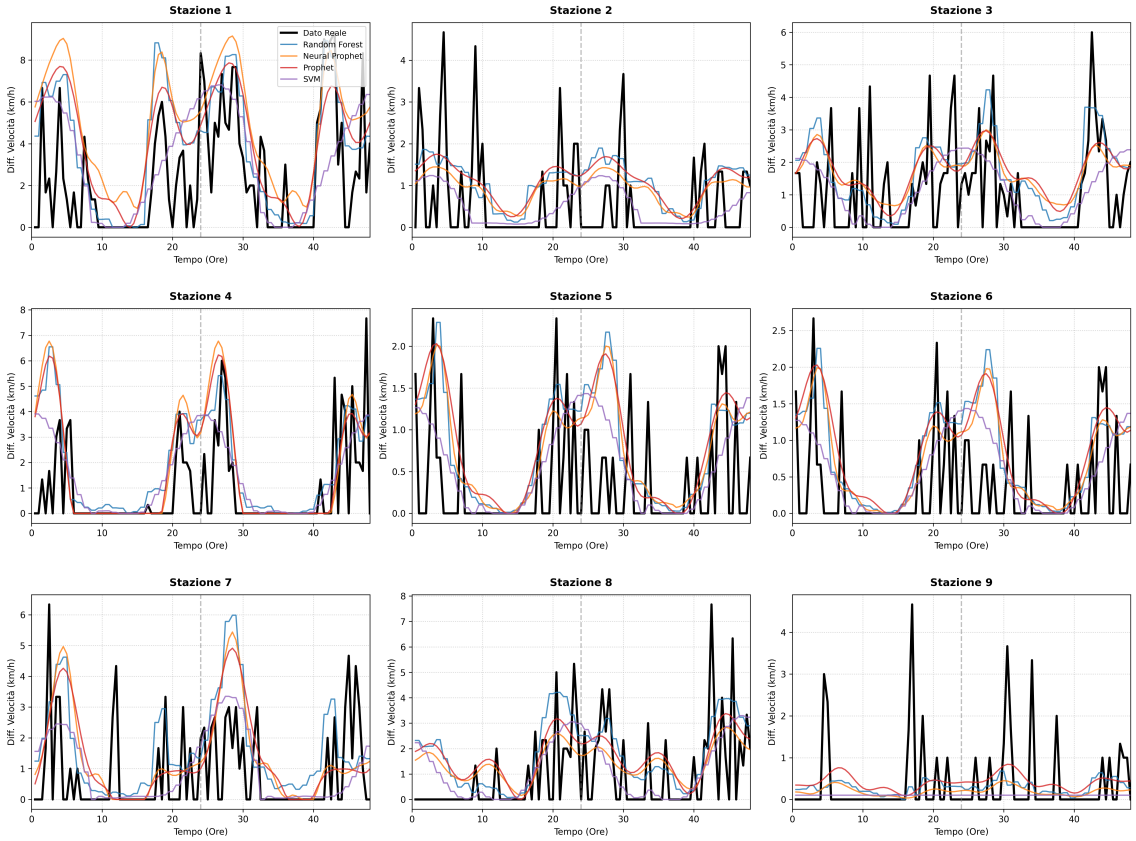


Figura 5.2: Sovrapposizione delle previsioni dei modelli sul dato reale (Δv) per le singole stazioni di monitoraggio (Scorrimento a 30 minuti)

L'ispezione dello scenario a 30 minuti conferma la capacità generale dei modelli di seguire le fluttuazioni del traffico. Tuttavia, il confronto qualitativo con lo scorrimento giornaliero non evidenzia un miglioramento sostanziale nella corrispondenza alla curva reale. Al contrario, alcuni algoritmi iniziano a mostrare segnali di ritardo nell'agganciare le variazioni. Risulta particolarmente rilevante l'assenza visiva dei modelli SARIMA e Stacked LSTM a causa dell'eccessiva richiesta di memoria RAM e dei tempi di *training* in mancanza di una GPU, forzandone l'esclusione tecnica da questo specifico scenario ad alta frequenza.

5.1.2 Valutazione quantitativa dell'errore (MAE e MAPE)

La valutazione quantitativa certifica una buona accuratezza raggiunta dalle architetture di base in entrambi gli scenari di scorrimento.

Scorrimento a 1 Giorno

La Figura 5.3 confronta l'accuratezza globale degli algoritmi calcolata su tutte le stazioni della rete stradale per lo scorrimento a 24 ore, riportando per ciascun modello l'errore medio assoluto (MAE) in km/h sulla sinistra e la deviazione percentuale relativa (MAPE) sulla destra.

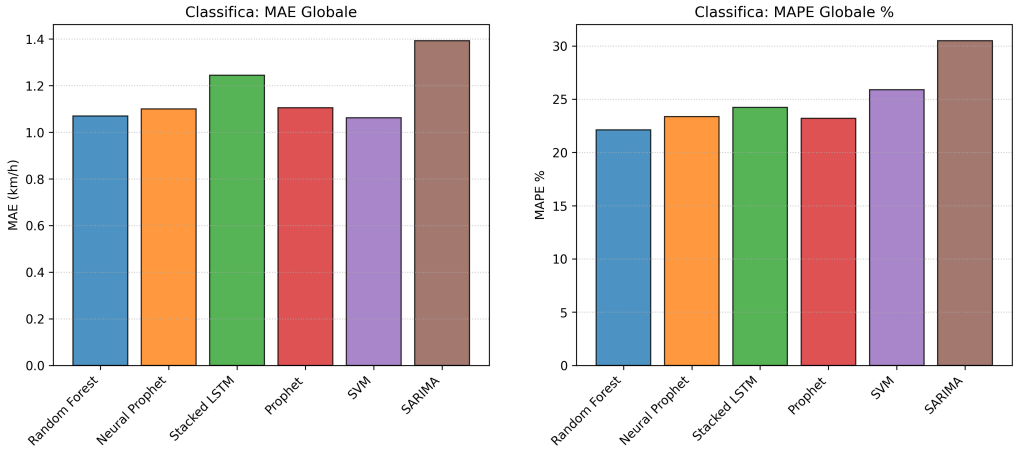


Figura 5.3: Confronto sulle metriche MAE (sinistra) e MAPE (destra) aggregate sull'intera rete (Scorrimento 1 giorno)

Dal grafico globale emerge che algoritmi capaci di mappare relazioni fortemente non lineari, come il Random Forest e SVM, si rivelano i più performanti, registrando un MAE prossimo a $1.05 - 1.10$ km/h. I modelli additivi (Prophet e Neural Prophet) mantengono prestazioni discrete (MAE ≈ 1.1 km/h). L'architettura neurale Stacked LSTM si posiziona su una fascia intermedia, con un errore assoluto intorno a 1.25 km/h e un MAPE del 24%, denotando una precisione inferiore ai metodi *ensemble*. Infine, il modello statistico SARIMA risulta il meno accurato, sfiorando gli 1.4 km/h e superando il 30% di scarto relativo.

Disaggregando il dato per singola stazione stradale (Figure 5.4 e 5.5), si nota come l'entità dell'errore sia legata alle caratteristiche della viabilità e ai flussi specifici di ciascun nodo stradale. La Stazione 1 presenta i picchi di errore più alti in senso assoluto (in particolare con SARIMA, che supera i 3.0 km/h di MAE), mentre la Stazione 9 registra gli scarti lineari minori (MAE intorno a 0.5 km/h), suggerendo una maggiore regolarità dei flussi in quell'area. Tuttavia, analizzando l'errore relativo (MAPE), stazioni a flusso mediamente più costante come la Stazione 2 mostrano percentuali di errore inferiori.

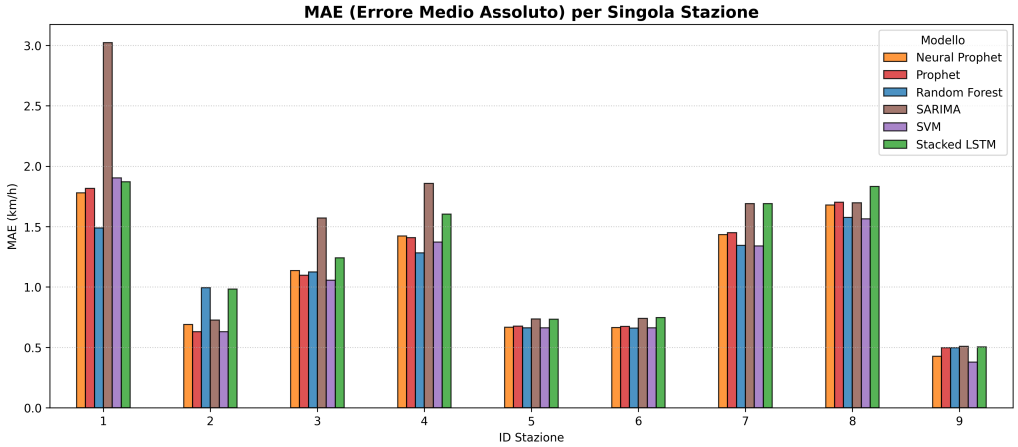


Figura 5.4: Errore Medio Assoluto (MAE) per singola stazione di monitoraggio (Scorrimento 1 giorno)

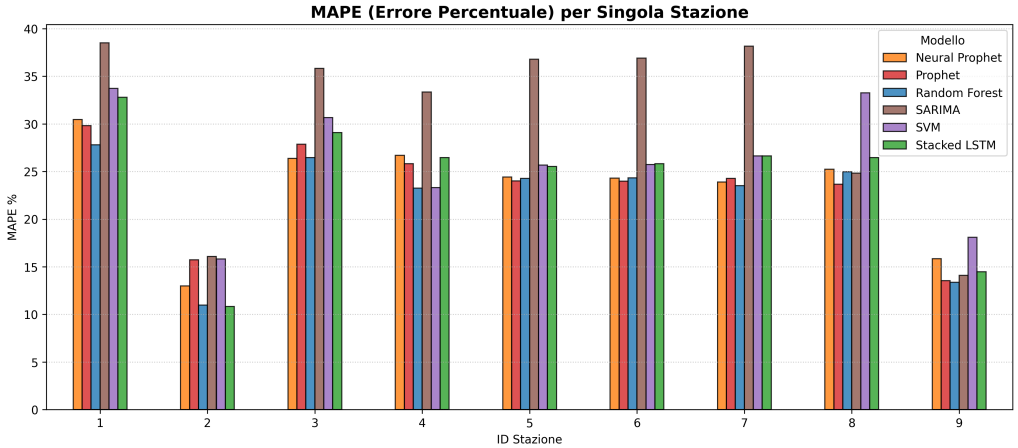


Figura 5.5: Errore Percentuale (MAPE %) per singola stazione di monitoraggio (Scorrimento 1 giorno)

Scorrimento a 30 minuti

I risultati aggregati per lo scenario ad alta frequenza (Figura 5.6) mostrano un quadro prestazionale essenzialmente identico, confermando che il riaddestramento continuo non produce vantaggi. SVM e Random Forest mantengono le performance migliori (MAE globale ancorato intorno a $1.05 - 1.10$ km/h).

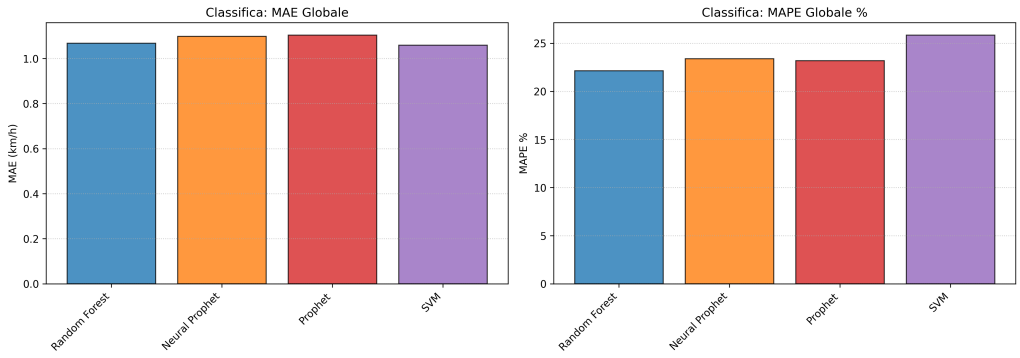


Figura 5.6: Confronto sulle metriche MAE (sinistra) e MAPE (destra) aggregate sull'intera rete (Scorrimento 30 minuti)

Disaggregando i risultati per stazione (Figure 5.7 e 5.8), i trend rispecchiano le medesime dinamiche osservate nello scenario giornaliero.

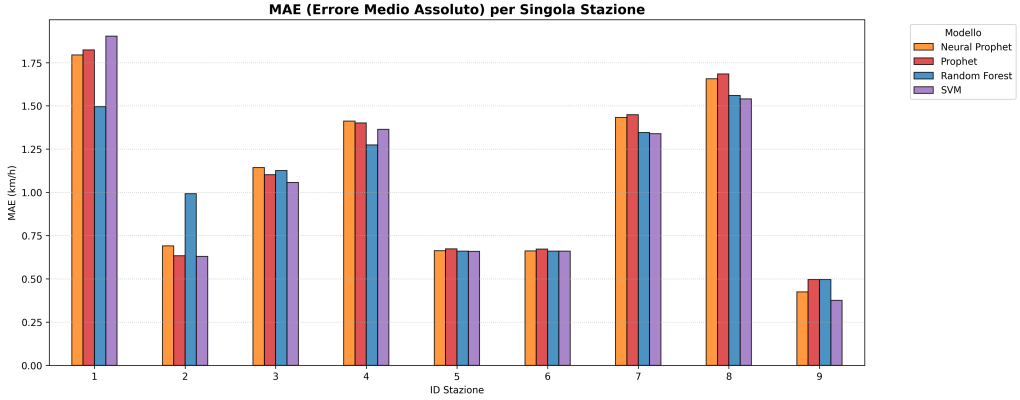


Figura 5.7: Errore Medio Assoluto (MAE) per singola stazione di monitoraggio (Scorrimento 30 minuti)

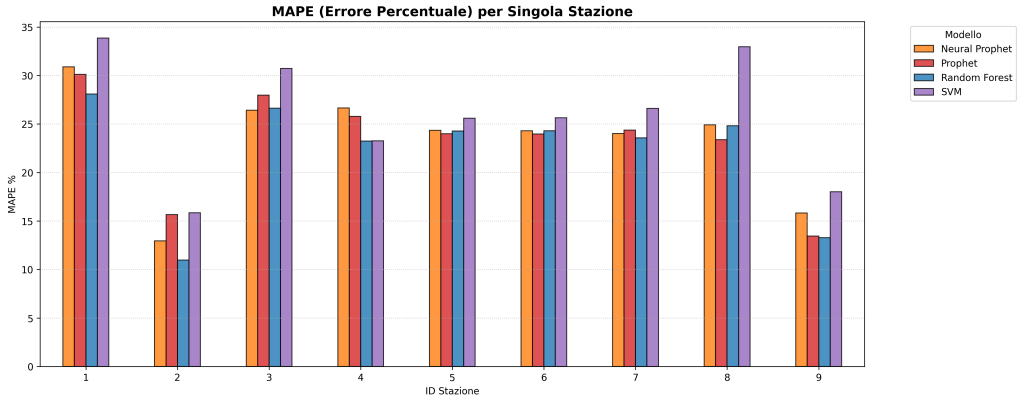


Figura 5.8: Errore Percentuale (MAPE %) per singola stazione di monitoraggio (Scorrimento 30 minuti)

Stabilità temporale dell'errore

Per valutare l'affidabilità del sistema nel tempo, le metriche sono state tracciate lungo

l'intero mese di test. Le Figure 5.9 e 5.10 illustrano l'andamento cronologico continuo dell'errore calcolato giorno per giorno nello scenario a 24 ore. Lo sfondo a bande verticali alternate permette di discernere visivamente l'impatto dei fine settimana rispetto alle fluttuazioni dei giorni lavorativi.

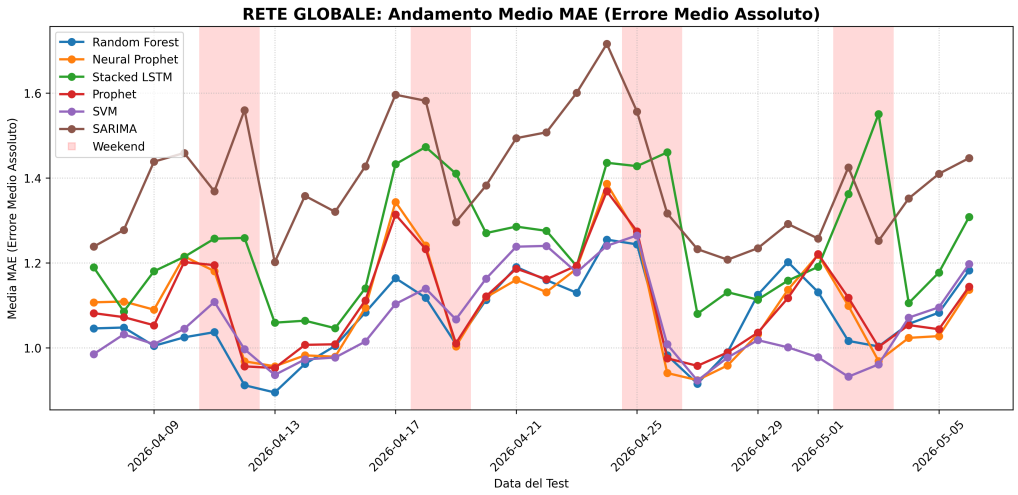


Figura 5.9: Andamento temporale del MAE aggregato a livello di rete (Scorrimento 1 giorno)

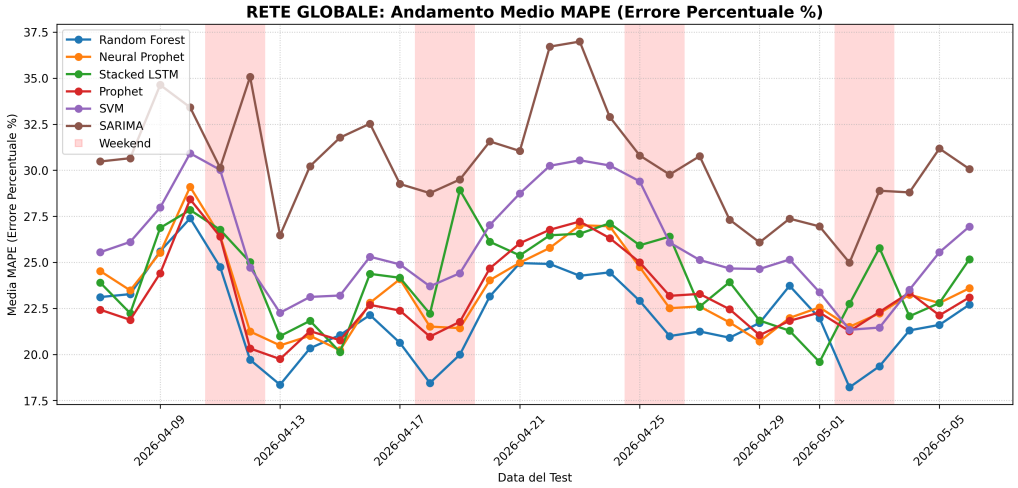


Figura 5.10: Andamento temporale del MAPE aggregato a livello di rete (Scorrimento 1 giorno)

La medesima analisi è proposta per lo scorrimento ogni 30 minuti (Figure 5.11 e 5.12).

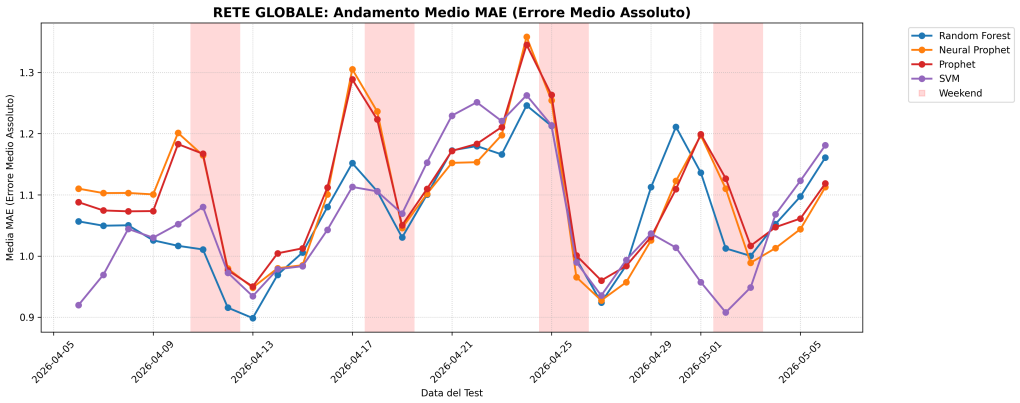


Figura 5.11: Andamento temporale del MAE aggregato a livello di rete (Scorrimento 30 minuti).

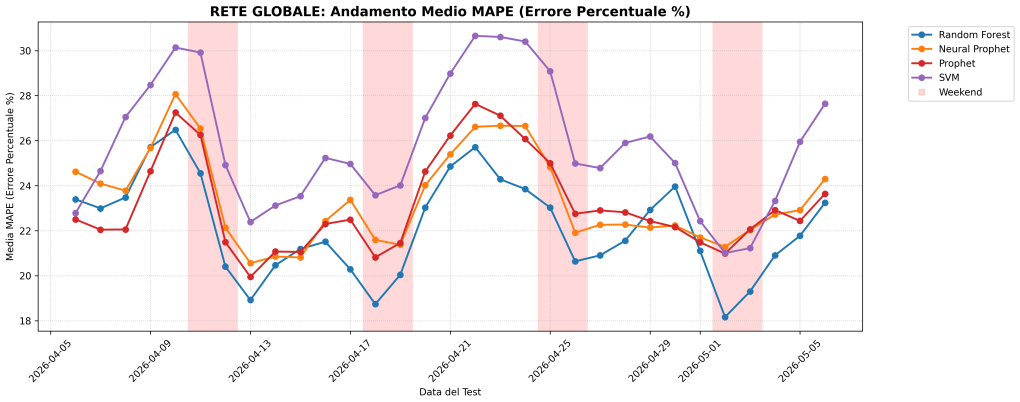


Figura 5.12: Andamento temporale del MAPE aggregato a livello di rete (Scorrimento 30 minuti).

Dai grafici si evince chiaramente come le prestazioni dei modelli non siano costanti, bensì influenzate dai cicli settimanali. In particolare, si osserva una netta diminuzione dell'errore in corrispondenza dei fine settimana (le bande verticali evidenziate in figura), quando i volumi di traffico si riducono e i flussi risultano meno soggetti a congestioni anomale. Al contrario, durante i giorni lavorativi, l'errore tende a innalzarsi a causa della maggiore instabilità e intensità dei picchi viabilistici.

Analizzando i risultati della Random Forest e dell'SVM, si nota una forte costanza prestazionale: l'errore medio globale si mantiene saldamente nel range di $0.9 - 1.7$ km/h. Si rilevano inoltre naturali flessioni dell'errore (visibili nelle bande rosa) in corrispondenza dei fine settimana, dove il traffico risulta ridotto e maggiormente regolare.

5.1.3 Decadimento dell'errore (orizzonte 48 ore) e confronto sliding window (24h e 30 min)

Un parametro critico per la valutazione di un sistema *Intelligent Transportation System* (ITS) è il mantenimento dell'affidabilità all'interno dell'orizzonte di *forecast*. La Figura

5.13 scompone il degrado temporale delle stime per stazione: sull'asse orizzontale scorre il tempo futuro (in ore), mentre sulle ordinate viene tracciato lo scostamento medio in km/h.

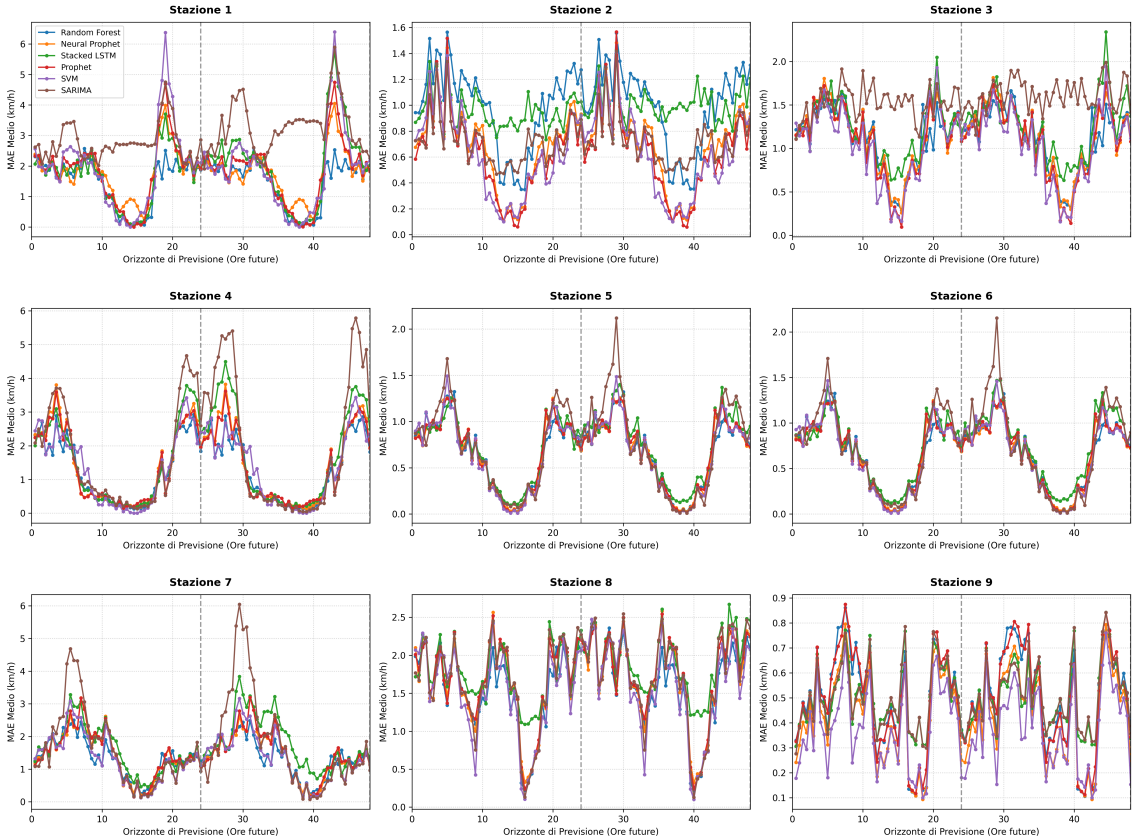


Figura 5.13: Grafici di decadimento dell'errore (MAE) all'aumentare dell'orizzonte di previsione (da 0 a 48 ore) per le singole stazioni (Scorrimento 1 giorno)

Nello scenario giornaliero, il decadimento dell'affidabilità non segue una retta monotona crescente, ma evidenzia un comportamento marcatamente ondulatorio in quasi tutte le stazioni. L'errore tende ad alzarsi in concomitanza con le ore di punta diurne (i momenti più volatili da prevedere), ma cala drasticamente in corrispondenza del passo a 24 e 48 ore (evidenziato dalle linee tratteggiate verticali). Questo dimostra l'elevata autocorrelazione del traffico a multipli di un giorno esatto: risulta infatti più agevole per i modelli prevedere il traffico di domani alla stessa ora, piuttosto che quello tra 8 ore.

L'equivalente grafico per lo scenario a 30 minuti è illustrato in Figura 5.14.

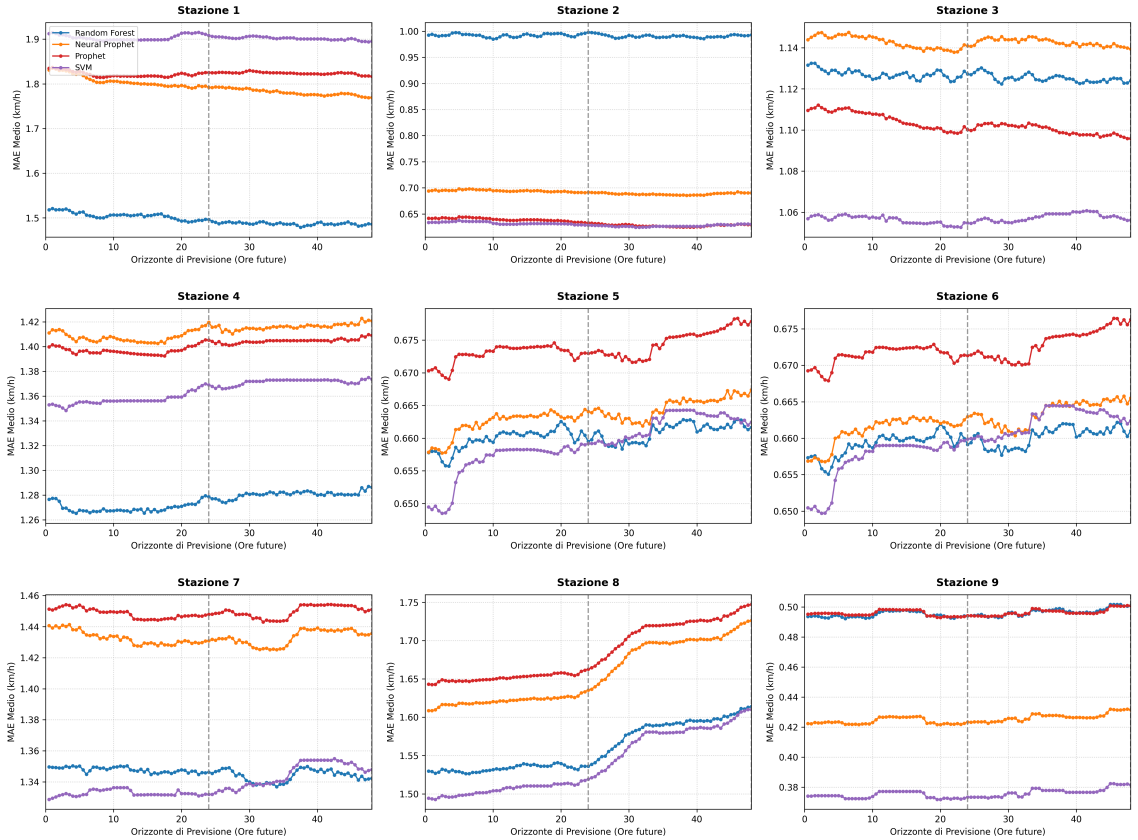


Figura 5.14: Grafici di decadimento dell'errore (MAE) all'aumentare dell'orizzonte di previsione (da 0 a 48 ore) per le singole stazioni (Scorrimento 30 minuti)

In questo caso, le stazioni centrali (es. Stazione 5 e 6) mantengono il decadimento a onda, mentre nodi con flussi più asimmetrici (come la Stazione 2 o la 4) mostrano una crescita più regolare dell'errore, sintomo di una minore volatilità locale intra-giornaliera.

Confronto Frequenze (24h e 30 min) e Costi Computazionali

Valutare il passaggio da uno scorrimento giornaliero (step 24h) a uno ad alta frequenza (step 30 min) impone un'analisi dei costi computazionali. I log di sistema estratti durante le fasi di collaudo forniscono un riscontro esplicito sul carico elaborativo sostenuto dall'infrastruttura.

L'esecuzione dell'intera pipeline a scorrimento giornaliero ha richiesto un tempo totale di circa 21 ore e 17 minuti (per un totale di 76.674 secondi operativi). Di contro, l'elaborazione del medesimo flusso dati con frequenza di aggiornamento a 30 minuti — pur depurata computazionalmente dai gravosi modelli SARIMA e LSTM — ha generato un'esplosione dei tempi di addestramento, superando i 5 giorni e 11 ore continuativi (474.265 secondi).

Le Figure 5.15 e 5.16 misurano l'impatto architetturale disaggregato per algoritmo: il blocco di sinistra espone l'onere computazionale (misurato in secondi) richiesto per l'aggiornamento dei pesi, mentre il blocco a destra quantifica la latenza applicativa online (in millisecondi), parametro essenziale per stabilire la fluidità del servizio in produzione.

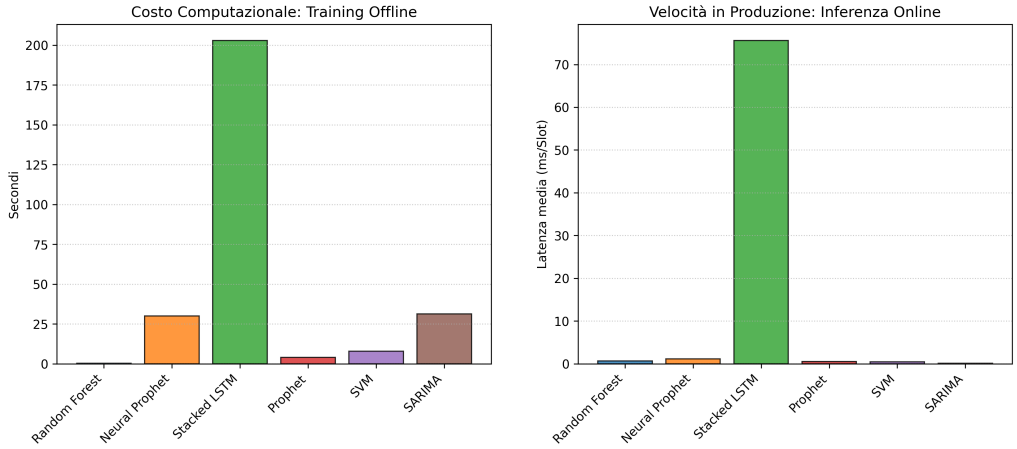


Figura 5.15: Costi computazionali dei modelli (Scorrimento 1 giorno): Tempo di addestramento offline in secondi (sinistra) e Latenza di inferenza online in millisecondi per slot (destra)

L'analisi in Figura 5.16 ripete la misurazione applicata al ciclo di addestramento previsto dallo scenario a 30 minuti.

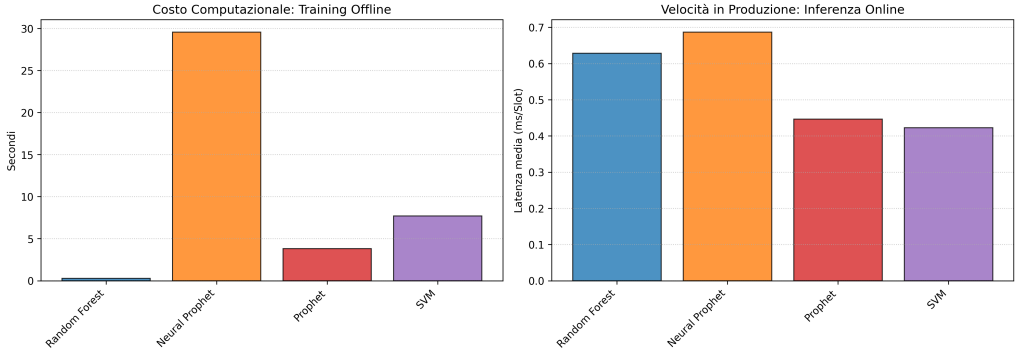


Figura 5.16: Costi computazionali dei modelli (Scorrimento 30 minuti): Tempo di addestramento offline in secondi (sinistra) e Latenza di inferenza online in millisecondi per slot (destra)

Considerando che le metriche di errore (MAE e MAPE) per lo scorrimento a 30 minuti non mostrano miglioramenti significativi rispetto allo scorrimento a 24 ore ed estromettono interamente SARIMA e LSTM per colli di memoria, è possibile trarre una conclusione metodologica: su una serie storica altamente periodica come il traffico, l'aggiornamento continuo dei pesi introduce un collo di bottiglia elaborativo ingiustificabile, dilatando i tempi di calcolo fino al quintuplo dell'orizzonte base. Richiedere al sistema di ricalcolare i pesi ogni 30 minuti risulterebbe dispendioso in un'architettura Smart City su vasta scala. Pertanto, l'addestramento *batch* giornaliero notturno rappresenta la scelta ingegneristica ottimale, garantendo latenze di inferenza in produzione misurabili in millisecondi e un errore assoluto saldamente confinato sotto la soglia di $1 - 2\ km/h$.

5.2 Risultati sul dataset delle colonnine EV

In questa sezione si valuta la traslazione dei modelli predittivi dal dominio continuo del traffico stradale a quello delle infrastrutture di ricarica. **La variabile target oggetto di previsione è il tasso di occupazione percentuale delle prese** presso le 7 stazioni target. L'obiettivo è verificare la resilienza e l'adattabilità dell'architettura di *forecasting*. Si noti che la Stazione 2, considerata nel dominio del traffico, è stata esclusa dall'infrastruttura di analisi EV in quanto non rappresentativa delle dinamiche di ricarica utili allo studio.

È opportuno precisare una scelta terminologica adottata nell'esposizione dei risultati. Mentre per il traffico stradale si fa riferimento allo specifico punto di rilevamento indicandolo come "Stazione" (es. Stazione 1), per l'infrastruttura di ricarica i termini "Colonnina" e "Zona" verranno utilizzati come sinonimi (es. Colonnina 1 equivale a Zona 1, dicitura quest'ultima mantenuta coerentemente all'interno dei grafici). Tale nomenclatura sottolinea come la singola colonnina monitorata sia di fatto la rappresentante dell'area geografica circostante (nel raggio di 150 metri) associata alla rispettiva stazione di rilevamento del traffico, garantendo l'esatta corrispondenza spaziale (Colonnina 1 = Zona 1 = Stazione 1).

5.2.1 Analisi visiva: sovrapposizione curve reali e previsioni

L'ispezione visiva delle previsioni evidenzia un limite strutturale del dominio. Le curve reali presentano un tipico andamento "a gradino" (*step function*), alternando ore di totale inutilizzo a repentine transizioni di saturazione dell'infrastruttura.

Scorrimento a 1 Giorno

La validazione grafica di questo comportamento è illustrata nella Figura 5.17, che mostra la sovrapposizione tra la curva reale (linea nera spessa) e le curve predette dai vari algoritmi per le 7 zone monitorate nello scenario ad aggiornamento giornaliero.

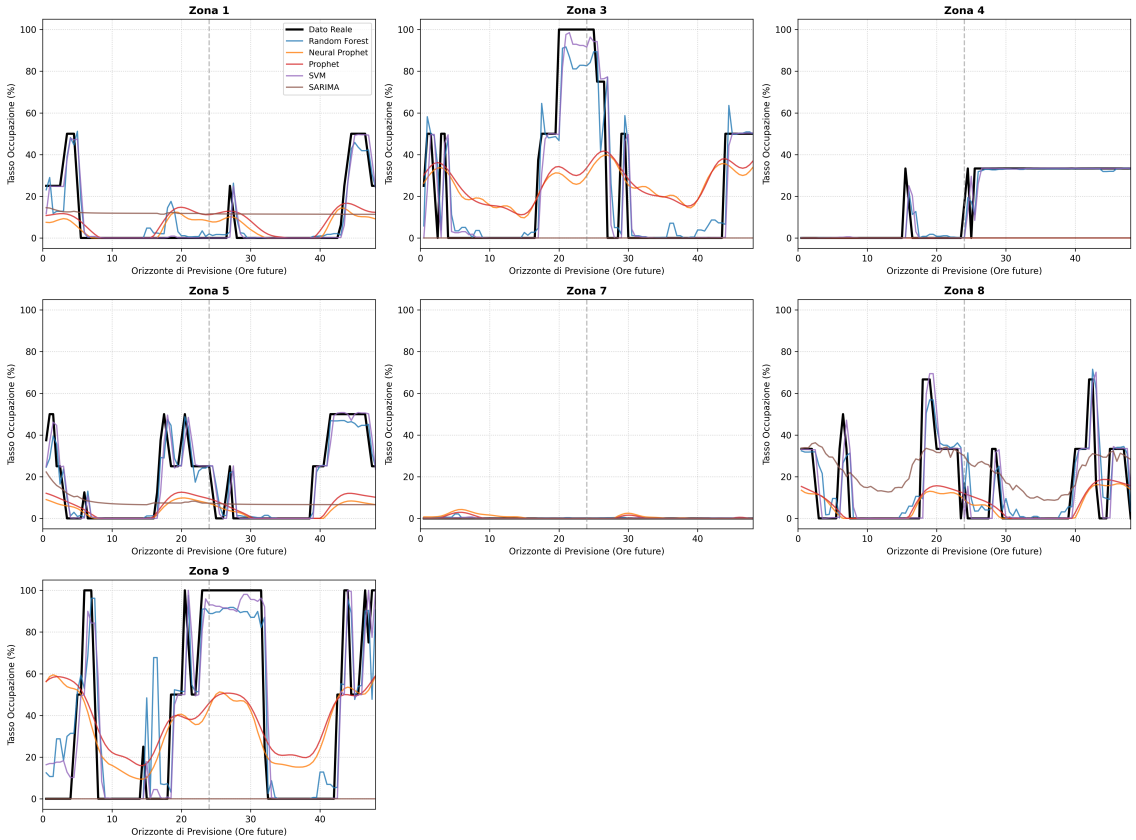


Figura 5.17: Grafici delle curve reali e predette per le 7 colonnine EV target (Scorrimento 1 giorno)

Come si evince dalla Figura 5.17, i modelli additivi (Prophet e Neural Prophet) e statistici lineari (SARIMA) faticano nell'agganciare le repentine variazioni di stato, producendo curve morbide e ondulate che si appiattiscono attorno al valore medio storico. Al contrario, i modelli basati su logiche non lineari, come Support Vector Machine (SVM) e Random Forest, mostrano una reattività superiore, riuscendo a innalzarsi ripidamente in corrispondenza del gradino imposto dalla curva reale (visibile ad esempio nelle colonnine 1, 5, 8 e 9).

Scorrimento a 30 minuti

L'estensione dell'ispezione visiva allo scenario ad alta frequenza (aggiornamento ogni 30 minuti, Figura 5.18) evidenzia lo stesso andamento dello scorrimento a 1 giorno. I profili delle previsioni ricalcano in modo sostanzialmente identico le imprecisioni registrate nello scenario a passo giornaliero.

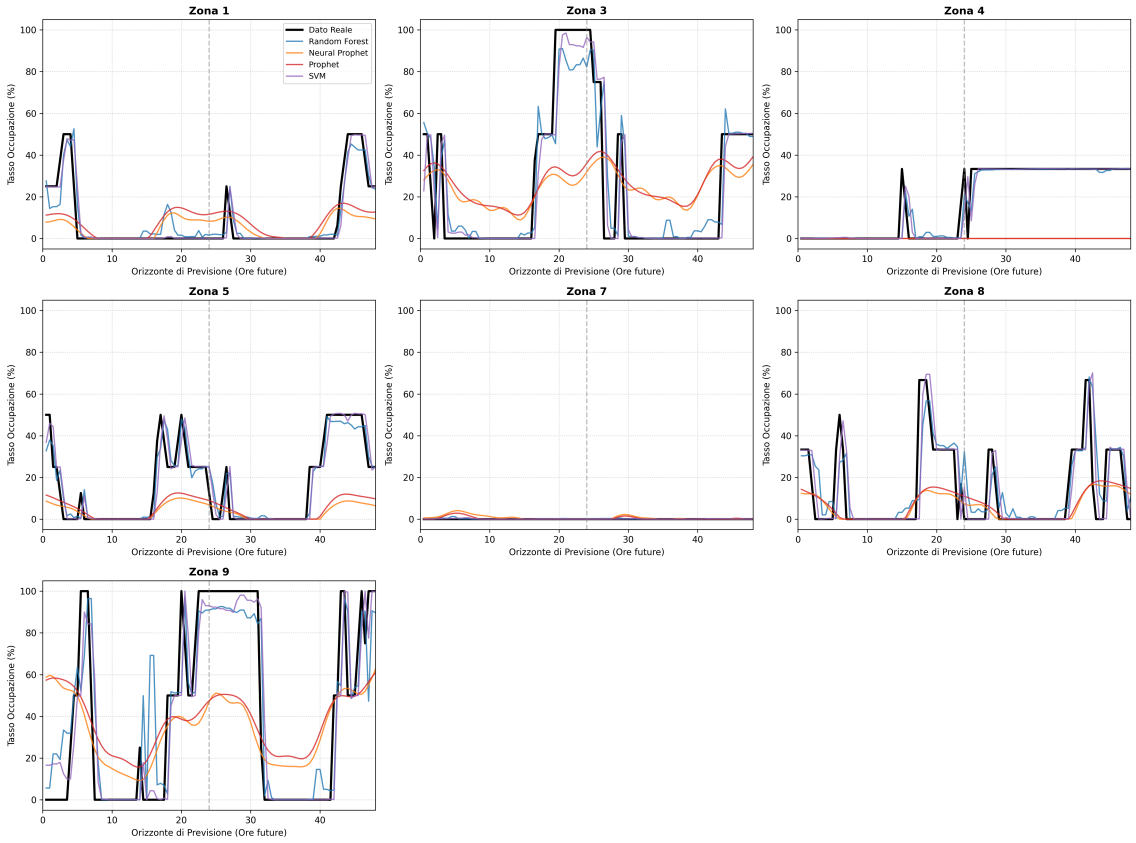


Figura 5.18: Grafici delle curve reali e predette per le 7 colonnine EV target (Scorrimento 30 minuti).

Questo comportamento ripetitivo evidenzia un limite strutturale: la natura discreta e impulsiva legata ai singoli eventi di ricarica [3] rende complessa la stima diretta tramite algoritmi progettati per estrapolare medie storiche continue.

5.2.2 Valutazione quantitativa e analisi di stabilità (media e varianza del MAE sui fold)

In questa sottosezione si espone l'analisi quantitativa dell'errore per entrambi gli scenari di test, valutando le performance medie dei modelli e la stabilità delle loro previsioni nel tempo.

Scorrimento a 1 giorno

La classifica globale dei modelli calcolata sull'intera infrastruttura a ciclo giornaliero è visualizzata nella Figura 5.19. Le barre più basse identificano i modelli SVM e Random Forest come i più accurati, seguiti dalle varianti Prophet e da SARIMA. Si noti che nell'esposizione grafica l'ordinamento differisce lievemente dalla visualizzazione tabellare, che segue l'ordine strettamente crescente del MAE globale.

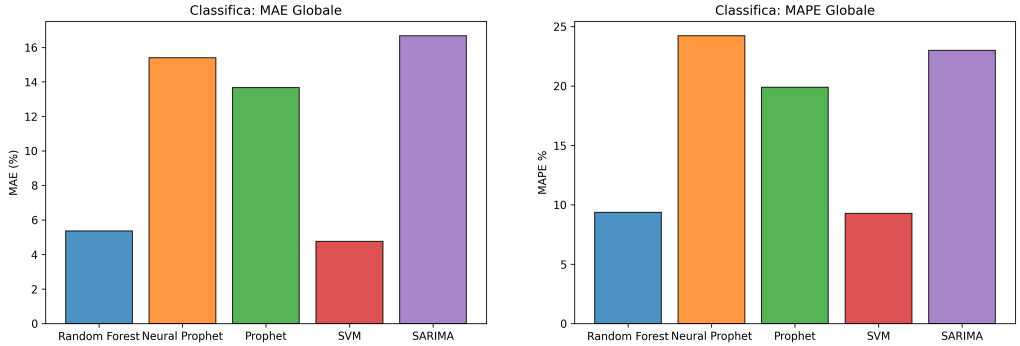


Figura 5.19: Classifica globale dei modelli predittivi sull'occupazione EV (Scorrimento 1 giorno): confronto sulle metriche MAE (sinistra) e MAPE (destra) aggregate

Risulta evidente un innalzamento degli scarti rispetto alla viabilità stradale. La Tabella 5.1 fornisce il dettaglio numerico estratto dalle predizioni. I modelli più performanti (SVM e Random Forest) contengono il MAE globale tra il 4.77% e il 5.37%, registrando un MAPE vicino al 9.3%. Le architetture additive come Neural Prophet e SARIMA registrano un degrado prestazionale con errori superiori al 15% e sfiorano il 24% di scarto relativo.

Modello	**MAE Globale [%]**	**MAPE Globale [%]**
SVM	4.77	9.29
Random Forest	5.37	9.38
Prophet	13.67	19.89
NeuralProphet	15.42	24.16
SARIMA	16.73	23.00

Tabella 5.1: Sintesi delle metriche di errore globali sul dataset delle colonnine EV (media aggregata sulle 7 colonnine attive per l'esperimento a 1 giorno)

In un contesto operativo di Smart City, un errore del 15-20% sull'occupazione percentuale si traduce in uno scarto pari a un'intera presa di ricarica mal calcolata per ogni colonnina, inficiando l'affidabilità del servizio (ad esempio indicando come "libera" una colonnina interamente saturata).

Considerando l'errore per singola colonnina di ricarica, si rileva un'alta varianza. La Figura 5.20 e la Figura 5.21 mostrano l'errore commesso in ciascuna colonnina. Nodi soggetti a picchi d'uso irregolari, come le colonnine 3 e 9, inducono i modelli a commettere macro-errori, con l'algoritmo SARIMA che in tali contesti arriva a superare il 35-40% di MAE (registrando fino a circa il 43% nella colonnina 9).

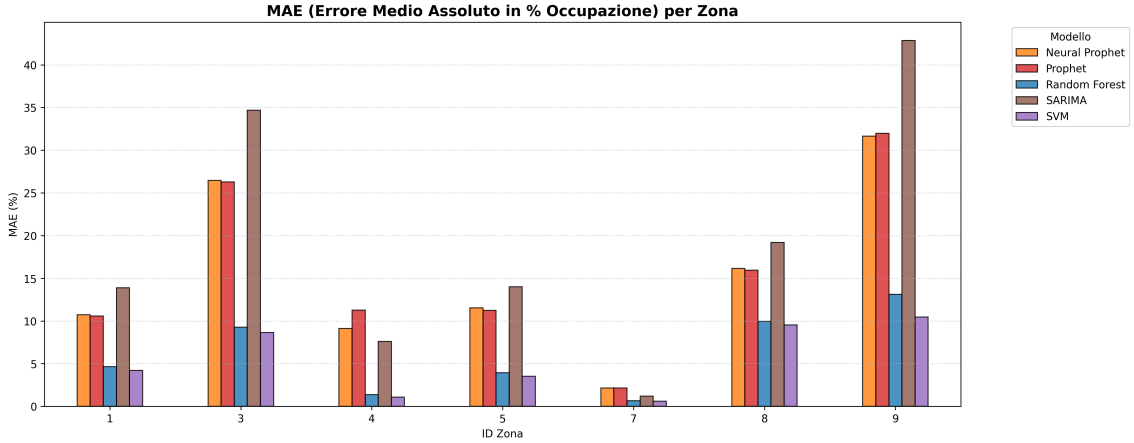


Figura 5.20: Errore Medio Assoluto (MAE espresso in % di occupazione) disaggregato per singola colonnina EV target (Scorrimento 1 giorno)

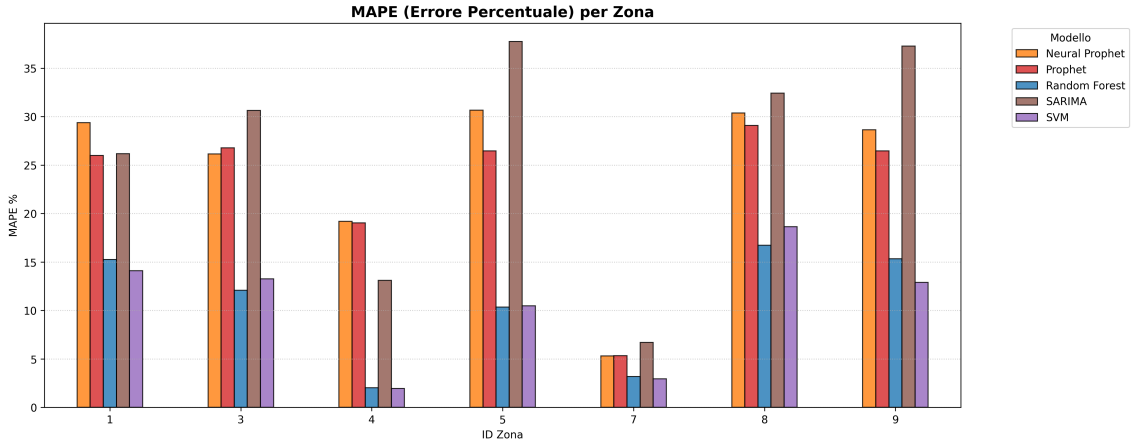


Figura 5.21: Errore Percentuale (MAPE %) disaggregato per singola colonnina EV target (Scorrimento 1 giorno)

L'analisi del decadimento temporale consolida il quadro di incertezza. I grafici a linee in Figura 5.22 tracciano l'evoluzione dell'errore (MAE) per i 48 slot semiorari futuri nello scenario giornaliero, aggregato sui 7 giorni di test per ogni colonnina. L'andamento instabile delle spezzate evidenzia come l'errore subisca picchi e flessioni legati all'ora del giorno.

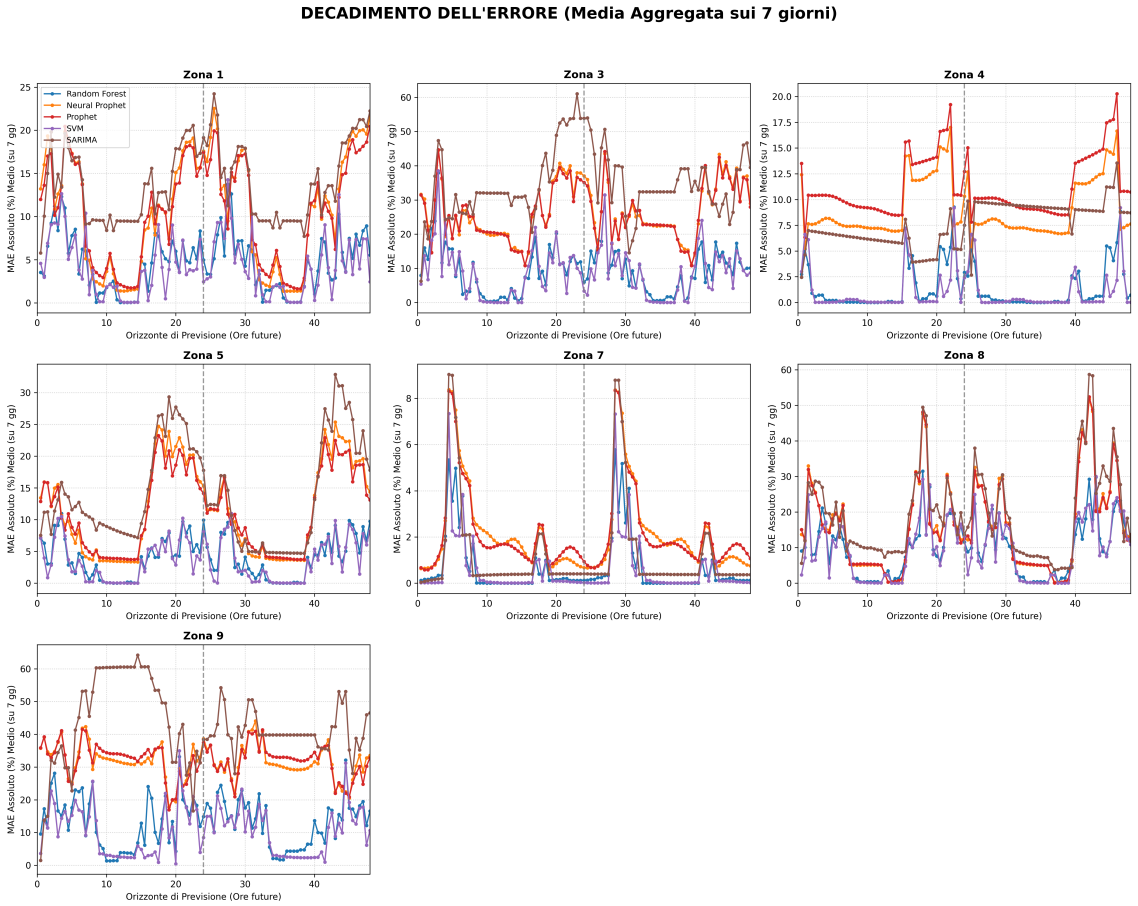


Figura 5.22: Grafici del decadimento dell'errore (MAE %) sull'occupazione EV per le colonnine target (Scorrimento 1 giorno, media aggregata sui 7 giorni).

Scorrimento a 30 minuti

L'analisi quantitativa dei test con frequenza a 30 minuti replica le problematiche del ciclo giornaliero, portando alla luce un fenomeno metodologico: **il riaddestramento continuo su un segnale ad alta varianza non migliora le performance predittive**.

L'istogramma globale in Figura 5.23 e i dati aggregati nella Tabella 5.2 dimostrano che il restringimento della finestra causa un lieve innalzamento dell'errore per tutti i modelli.

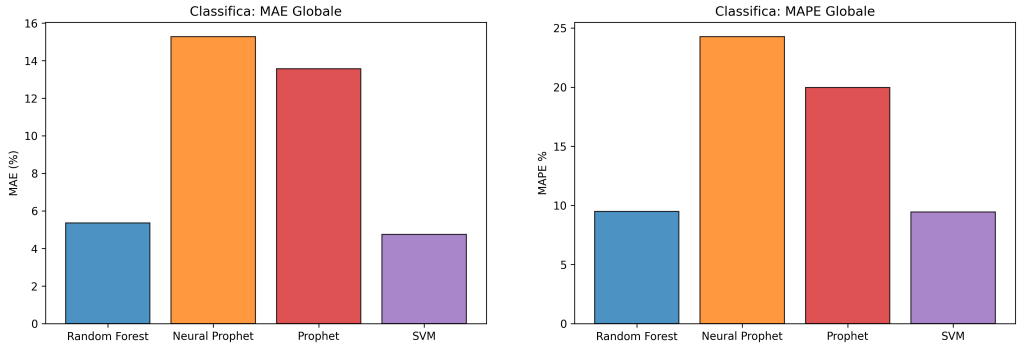


Figura 5.23: Classifica globale dei modelli predittivi sull'occupazione EV (Scorrimento 30 minuti): confronto sulle metriche MAE (sinistra) e MAPE (destra) aggregate

Modello	MAE Globale [%]	MAPE Globale [%]
SVM	5.44	10.80
Random Forest	6.13	10.83
NeuralProphet	15.28	24.28
Prophet	15.51	22.82

Tabella 5.2: Sintesi delle metriche di errore globali sul dataset delle colonnine EV (media aggregata sulle 7 colonnine attive per l'esperimento a 30 minuti)

Come si nota dalla tabella, il MAE del Random Forest sale dal 5.37% al 6.13%, e quello della SVM passa dal 4.77% al 5.44%. Ricalcolare i pesi ogni 30 minuti su un segnale così impulsivo genera *overfitting* sul rumore recente, destabilizzando il modello invece di aiutarlo a prevedere il gradino strutturale (i modelli SARIMA e LSTM sono stati esclusi a causa dei limiti computazionali e di memoria riscontrati in questa specifica frequenza).

Scorporando i dati localmente (Figure 5.24 e 5.25), i picchi di errore restano anche qui in corrispondenza delle colonnine 3 e 9, dimostrando che l'incapacità di estrapolare la varianza non dipende dalla frequenza di aggiornamento, ma dalla caoticità intrinseca della zona.

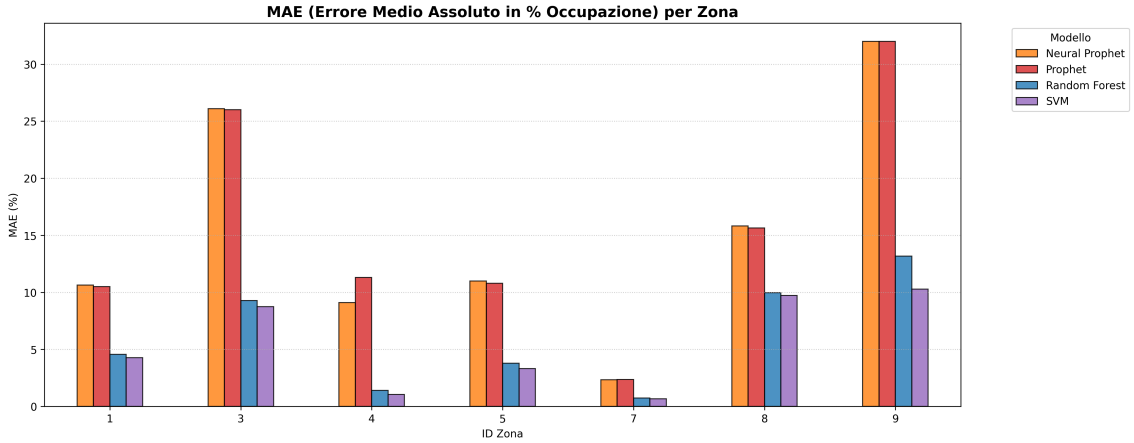


Figura 5.24: Errore Medio Assoluto (MAE %) disaggregato per singola colonnina EV target (Scorrimento 30 minuti). La colonnina 2 è esclusa dall'analisi.

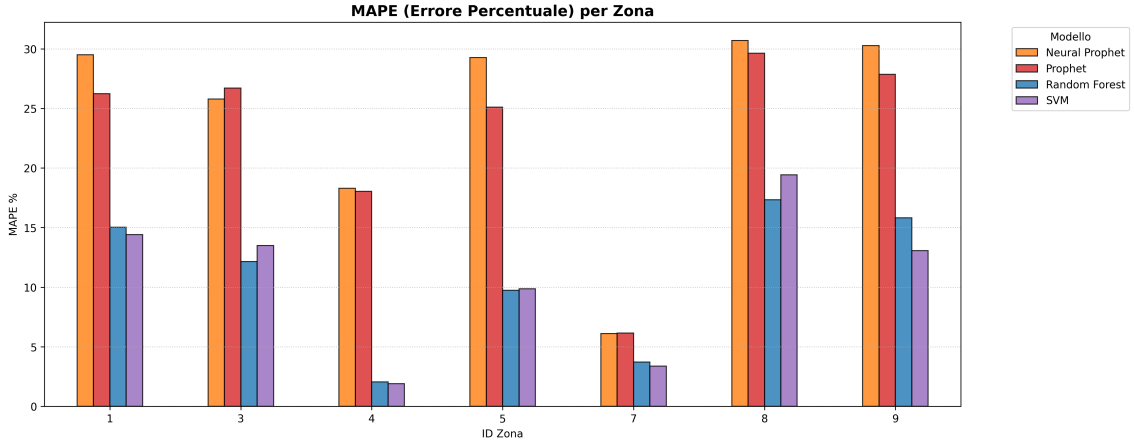


Figura 5.25: Errore Percentuale (MAPE %) disaggregato per singola colonnina EV target (Scorrimento 30 minuti). La colonnina 2 è esclusa dall'analisi.

Infine, l'invarianza predittiva trova conferma nel tracciamento del decadimento nel tempo per tutte le colonnine di ricarica, aggregato sulle 336 finestre (Figura 5.26).

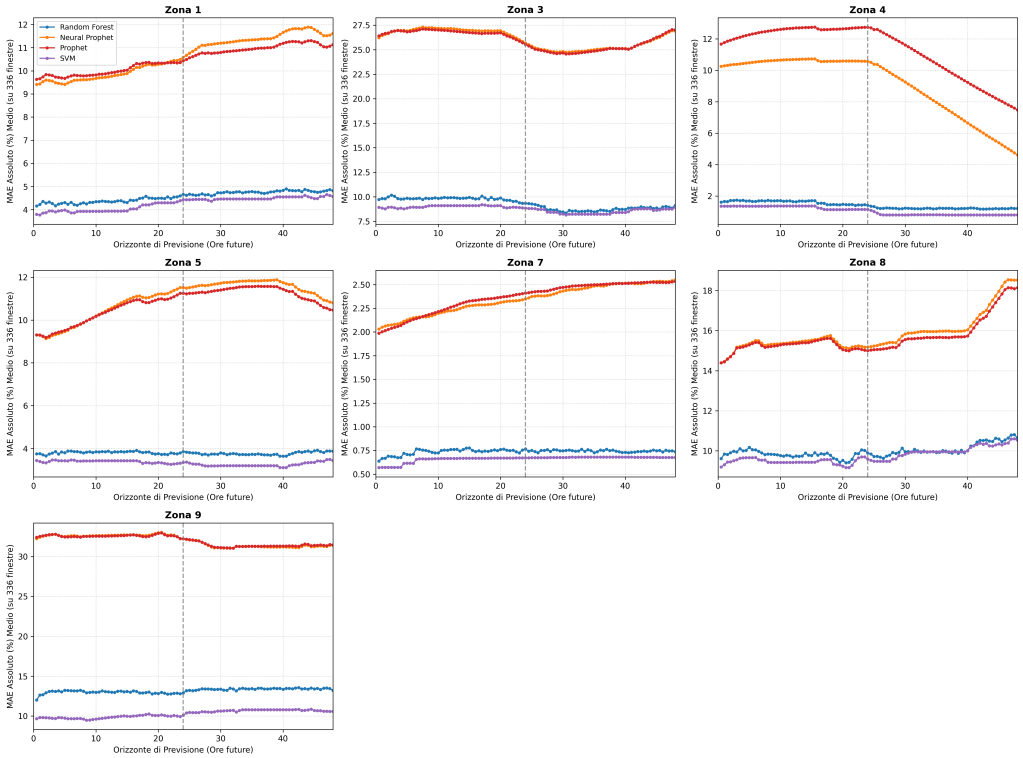


Figura 5.26: Grafici del decadimento dell'errore (MAE %) sull'occupazione EV per le colonnine target (Scorrimento 30 minuti, media aggregata su 336 finestre)

L'aumento dell'orizzonte di previsione produce un decadimento marcatamente monotono (crescita graduale o plateau). A differenza dell'andamento fortemente oscillante registrato nello scorrimento a 1 giorno (dove l'orizzonte coincide con le ore del giorno), l'ag-

giornamento a 30 minuti distribuisce le misurazioni sull'intero ciclo circadiano settimanale, portando le curve di errore a perdere la componente periodica diurna/notturna.

5.2.3 Impatto della frequenza di sliding window (step 24h e step 30 min)

L'indagine sistematica tra i due approcci algoritmici consente di delineare una risposta ingegneristica definita: nel dominio dell'infrastruttura di ricarica EV, incrementare la frequenza di *Sliding Window* allo scopo di ottenere un *nowcasting* a 30 minuti (ovvero una previsione a brevissimo termine che proietta lo stato attuale nell'immediato futuro) non produce alcun vantaggio operativo.

Come dimostrato dall'ispezione delle curve e dall'analisi delle metriche di errore globale (Tabelle 5.1 e 5.2), lo scorrimento a 30 minuti non permette ai modelli classici di anticipare le transizioni fisiche. Inoltre, il continuo aggiornamento spinge gli algoritmi a inseguire il rumore statistico di breve termine, innalzando lievemente l'errore percentuale.

A questa inefficacia predittiva si contrappone un elevato onere computazionale, dettagliato nelle Figure 5.27 e 5.28, che illustrano rispettivamente i tempi di *training* offline (a sinistra) e le latenze di inferenza online (a destra) per i due scenari.

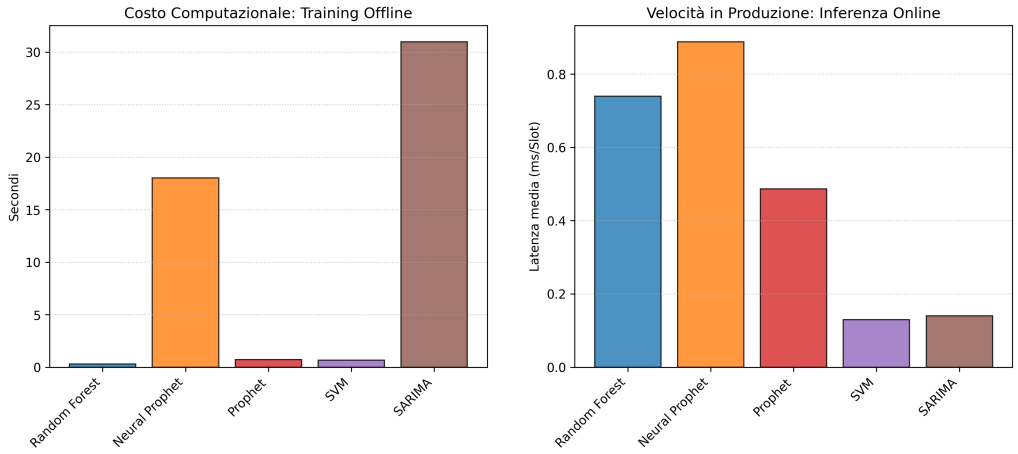


Figura 5.27: Costi computazionali dei modelli sull'occupazione EV (Scorrimento 1 giorno): Tempo di addestramento offline in secondi (sinistra) e Latenza di inferenza online in millisecondi per slot (destra)

I log di sistema certificano che l'esecuzione dell'intera pipeline a scorrimento giornaliero (7 iterazioni) richiede circa 36 minuti totali. Di questi, come visibile dal picco nel grafico, quasi 20 minuti sono assorbiti dal solo algoritmo SARIMA: la natura "a gradino" e non stazionaria del segnale EV impedisce la rapida convergenza dei parametri statistici classici, costringendo l'ottimizzatore interno a reiterare infruttuosamente le iterazioni.

Lo scenario ad alta frequenza fa esplodere ulteriormente i tempi di calcolo.

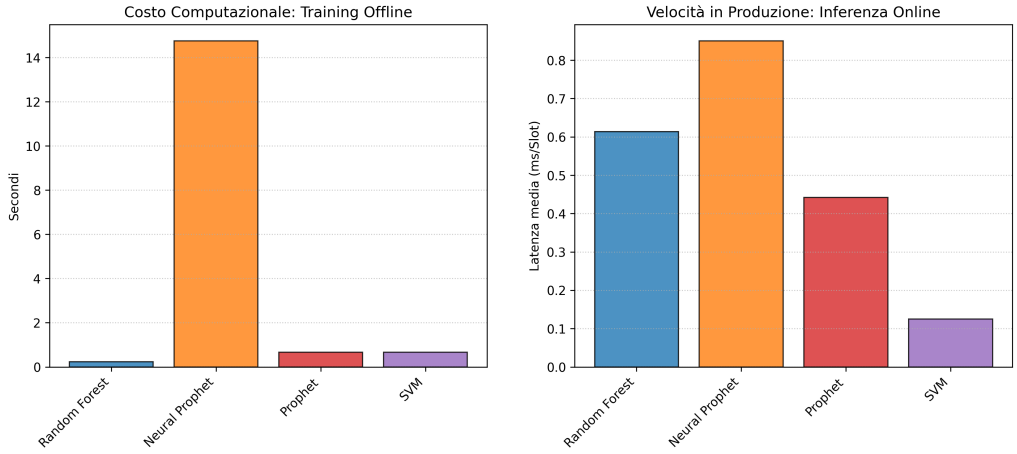


Figura 5.28: Costi computazionali dei modelli sull'occupazione EV (Scorrimento 30 minuti): Tempo di addestramento offline in secondi (sinistra) e Latenza di inferenza online in millisecondi per slot (destra). SARIMA e LSTM sono esclusi.

L'addestramento e il test con *step* a 30 minuti (336 iterazioni, pur escludendo i gravosi modelli SARIMA e LSTM) hanno richiesto ben 9 ore, 20 minuti e 37 secondi complessivi sul server dipartimentale. Sebbene l'inferenza online rimanga estremamente rapida per tutti i modelli valutati (costantemente sotto la frazione di millisecondo per slot), l'aggiornamento continuo dei pesi in fase di addestramento si rivela proibitivo.

Alla luce dell'elevato onere computazionale richiesto per il ricalcolo continuo in produzione, a fronte di un lieve peggioramento in termini di accuratezza, il ricorso al *forecasting* classico sull'occupazione puntuale risulta un approccio inadeguato, validando l'ipotesi di ripiegare sulla ricerca di predittori indiretti fortemente correlati, come l'analisi del traffico stradale limitrofo. Sotto il profilo sistemistico, l'approccio computazionale più efficiente per l'intera architettura ITS consiste in un addestramento eseguito in modalità *batch* con cadenza giornaliera (tipicamente nelle ore notturne), sfruttando l'estrema rapidità dell'inferenza per fornire le stime continuative durante il giorno.

5.3 Studio di correlazione statistica

Alla luce delle criticità emerse nell'applicazione dei modelli di *forecasting* classico sulle dinamiche a gradino e ad alta variabilità delle infrastrutture EV, questa sezione esplora l'ipotesi centrale dell'elaborato: verificare se la congestione stradale (Δv) possa agire da indicatore indiretto per l'impegno della rete di ricarica. A tale scopo, le due serie temporali storiche (Traffico e Occupazione EV) sono state allineate temporalmente tramite *resampling* con granularità di 1 ora. Tale scelta metodologica, differente rispetto al campionamento semiorario utilizzato per l'addestramento dei modelli predittivi, si è resa necessaria per regolarizzare il rumore statistico di brevissimo periodo e favorire l'estrazione di trend macroscopici tra due segnali di natura fisica dissimile, filtrando esclusivamente le fasce orarie con rilevazioni contemporanee in prossimità spaziale (raggio di 150 metri).

5.3.1 Analisi di correlazione spazio-temporale

L'analisi quantitativa iniziale è stata condotta avvalendosi di due metriche statistiche classiche [20, 21]: l'indice di Pearson, per valutare la dipendenza lineare, e l'indice di Spearman (basato sui ranghi), particolarmente indicato per confrontare relazioni monotone tra un segnale continuo (il traffico) e una variabile discreta "a gradino" (l'occupazione). Entrambi gli indici restituiscono un valore compreso tra -1 e 1.

La Tabella 5.3 riassume i coefficienti calcolati in modo sincrono (Lag 0), corredati dai rispettivi P-Value per certificarne la significatività statistica ($\alpha = 0.05$).

ID Stazione	**Pearson**	**P-Value (P)**	**Spearman**	**P-Value (S)**
1	0.518	< 0.001	0.569	< 0.001
3	0.189	0.008	0.236	0.001
4	0.109	0.126	0.147	0.040
5	0.186	0.009	0.197	0.006
7	0.235	0.001	0.141	0.048
8	0.387	< 0.001	0.422	< 0.001
9	0.074	0.302	0.041	0.565

Tabella 5.3: Risultati dell'analisi di correlazione sincrona (Lag 0) tra traffico (Δv) e occupazione colonnine. Un P-Value < 0.05 indica significatività statistica

I risultati dimostrano l'esistenza di una dipendenza statistica positiva e significativa in 6 stazioni su 7, sebbene di intensità eterogenea. La relazione risulta **forte** unicamente per la Stazione 1 ($\rho = 0.569$) e **moderata** per la Stazione 8 ($\rho = 0.422$), mentre le restanti stazioni statisticamente significative (3, 4, 5 e 7) mostrano correlazioni **deboli**, comprese tra 0.14 e 0.24. Fa eccezione la Stazione 9, per cui il test non rifiuta l'ipotesi di assenza di correlazione (P-Value = 0.565). Come prevedibile per il confronto con variabili a gradino, l'indice di Spearman restituisce in tutti i casi valori sistematicamente migliori rispetto a Pearson, confermando che la relazione è monotona (all'aumentare di una variabile aumenta l'altra) ma non strettamente lineare.

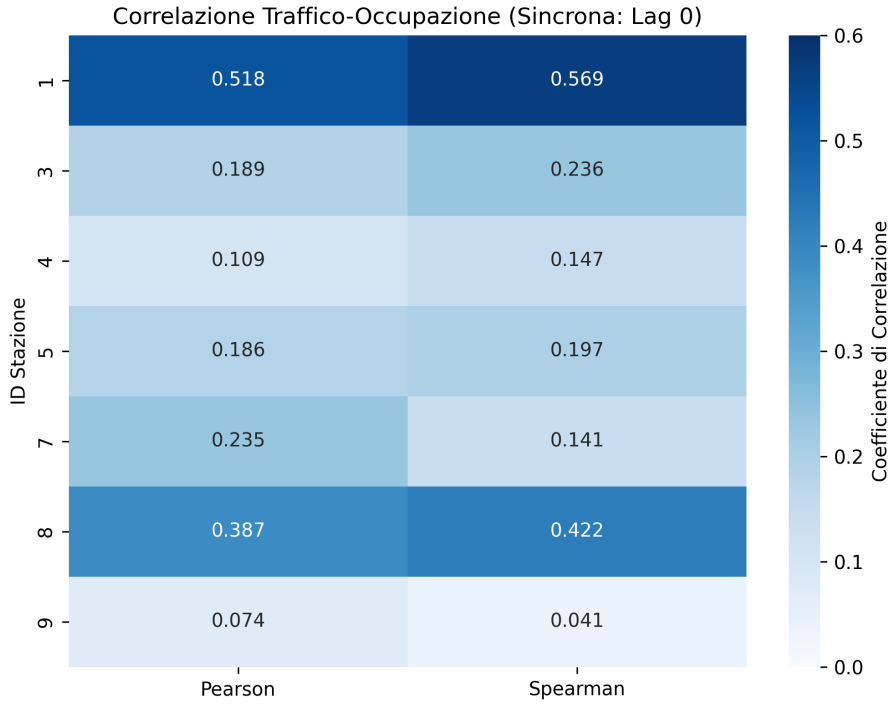


Figura 5.29: Heatmap dei coefficienti di correlazione di Pearson e Spearman (Lag 0) per singola stazione

Per confermare visivamente la natura di questa relazione, la Figura 5.30 illustra lo *scatterplot* tra la congestione veicolare (Δv) sull'asse delle ascisse e l'occupazione EV sull'asse delle ordinate per le Stazioni 1, 5 e 8. La linea di regressione rossa (con la relativa banda di confidenza) mostra una chiara tendenza crescente, più marcata per le Stazioni 1 e 8 e più tenue per la Stazione 5, coerentemente con i rispettivi coefficienti di Spearman. All'aumentare del traffico locale, la percentuale di occupazione delle colonnine tende a salire.

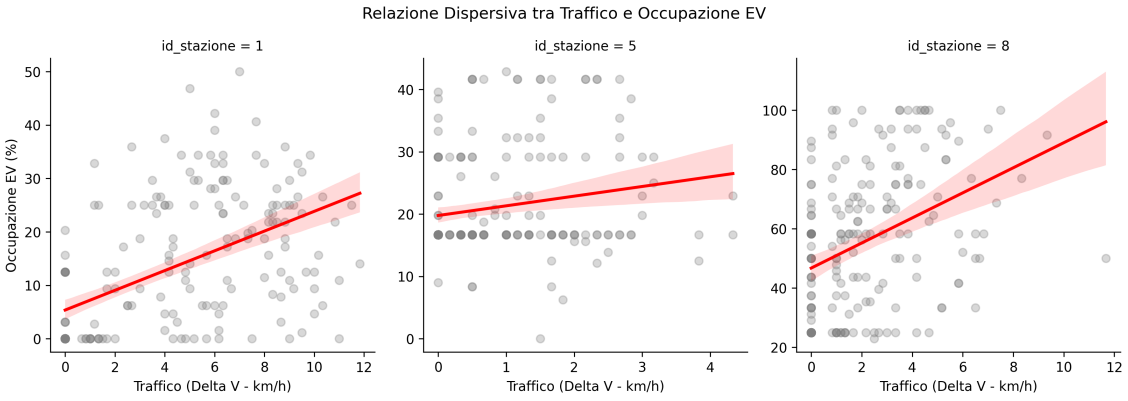


Figura 5.30: Relazione dispersiva tra traffico e occupazione EV per le stazioni 1, 5 e 8

Disaggregazione per contesto: feriale e weekend

Per comprendere a fondo le dinamiche di utilizzo, la correlazione globale è stata disaggre-

gata analizzando separatamente i giorni lavorativi (lunedì-venerdì) e i giorni festivi (sabato e domenica). La Figura 5.31 rivela dinamiche urbane eterogenee.

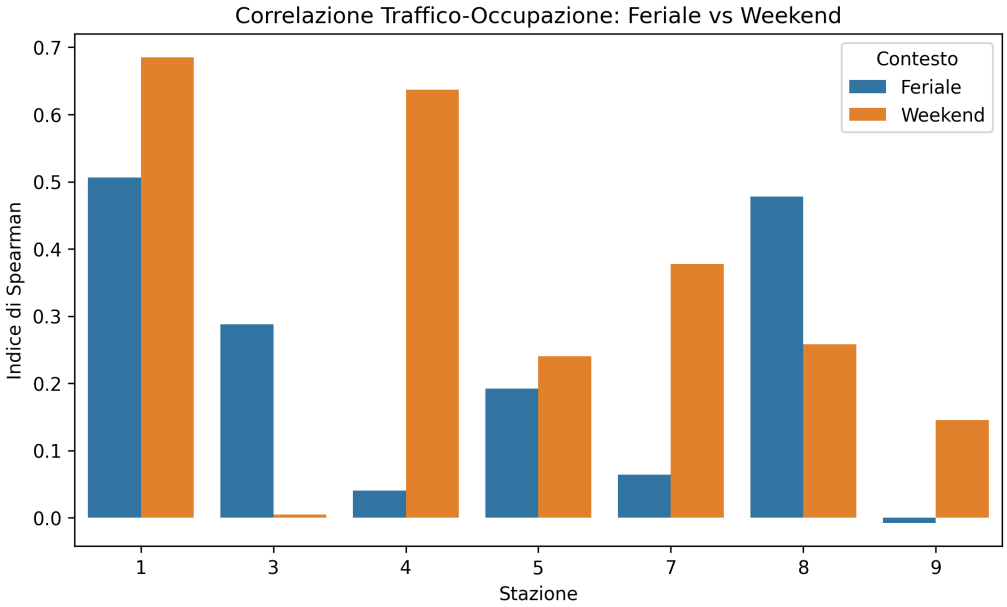


Figura 5.31: Correlazione di Spearman tra traffico e occupazione EV disaggregata per contesto (Feriale e Weekend)

Per stazioni come la 3 e la 8, la correlazione è decisamente più marcata nei giorni feriali (es. 0.478 nella Stazione 8 contro 0.258 nel weekend), suggerendo un'utenza pendolare o lavorativa strettamente legata ai flussi di traffico feriali. Al contrario, stazioni come la 1, la 4 e la 7 mostrano una correlazione molto più alta durante i fine settimana (fino a 0.685 per la Stazione 1 e 0.637 per la 4), indicando verosimilmente un utilizzo legato alla mobilità residenziale o all'afflusso verso le vicine aree di svago e vita notturna nei giorni di riposo, dove i picchi di traffico coincidono fortemente con i picchi di ricarica. Fanno eccezione nodi specifici come la Stazione 9, la quale non presenta alcuna correlazione rilevante nel periodo feriale (-0.008), confermando che l'impegno di quella specifica colonnina è guidato da fattori non viabilistici, probabilmente legati a ricariche notturne residenziali.

Cross-correlazione temporale (time-lag)

Tuttavia, il reale potenziale predittivo risiede nella consequenzialità temporale degli eventi: l'utente genera congestione stradale mentre guida verso una destinazione, e solo successivamente collega il veicolo alla colonnina. Per dimostrare questo fenomeno, è stata eseguita un'analisi di cross-correlazione introducendo ritardi temporali artificiali (*time lag*) di 1, 2 e 3 ore sull'asse dell'occupazione EV (Figura 5.32).

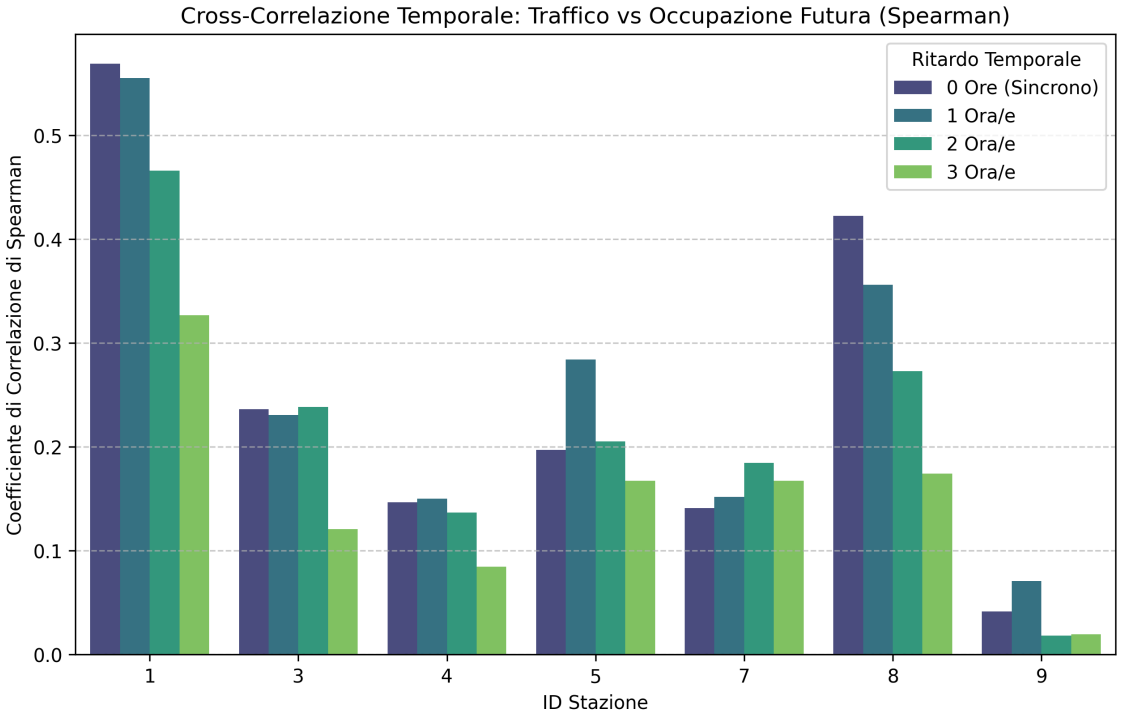


Figura 5.32: Analisi di cross-correlazione di Spearman con *time-lag* da 0 a 3 ore

I dati di estrapolazione temporale confermano l'ipotesi in molteplici snodi nevralgici. Per stazioni come la 5, la correlazione sincrona (Lag 0) pari a 0.197 sale a 0.284 applicando un ritardo di 1 ora. Ciò suggerisce che l'incremento di traffico in quella zona anticipa un conseguente aumento dell'occupazione EV entro i successivi 60 minuti. Dinamiche analoghe si registrano nella Stazione 7, dove la correlazione massima (0.184) si ottiene imponendo 2 ore di ritardo.

5.3.2 Validazione del traffico come predittore indiretto

Per validare matematicamente l'ipotesi che il traffico non sia solo "associato" ma agisca fisicamente come predittore dell'occupazione, è stato condotto il **Test di Causalità di Granger** [22]. Questo test statistico verifica se i valori passati di una serie temporale (il traffico) contengono informazioni utili e significative per prevedere i valori futuri di un'altra serie (l'occupazione EV).

Il test (impostato con un livello di significatività $\alpha = 0.05$) è stato applicato alle stazioni più rappresentative per distribuzione del traffico (1, 5 e 8) e ha restituito risultati determinanti:

- **Stazione 1:** Il test conferma la causalità predittiva del traffico sull'occupazione EV per orizzonti di 1, 2 e 3 ore nel futuro (tutti i P-Value ≤ 0.0041).
- **Stazione 5:** Il test conferma la causalità predittiva per tutti e tre i lag temporali analizzati, con una significatività massima a 2 ore di distanza (P-Value =

0.0031). Tale scostamento rispetto al picco di cross-correlazione (registrato a 1 ora) è del tutto coerente: mentre Spearman valuta il mero allineamento geometrico dei picchi, il test di Granger certifica in quale preciso intervallo temporale il segnale viabilistico massimizza il proprio apporto di *informazione predittiva* per il futuro dell'occupazione.

- **Stazione 8:** Il test non ha rifiutato l'ipotesi nulla (P-Value > 0.05), indicando che per questa specifica stazione, pur essendoci un'elevata correlazione sincrona, la storia del traffico locale non migliora statisticamente la previsione futura dell'occupazione.

I risultati dei test di Granger, uniti ai coefficienti di correlazione estratti in precedenza, costituiscono la svolta ingegneristica per la progettazione architetturale dello studio.

Come emerso dai test, vi è una marcata differenza di accuratezza ottenuta applicando gli stessi modelli predittivi ai due domini. Entrambi i fenomeni analizzati possiedono un'intrinseca componente aleatoria: anche il traffico, infatti, è costantemente influenzato da fattori incerti quali incidenti, condizioni meteorologiche e interazioni imprevedibili tra veicoli. Tuttavia, il traffico stradale (Δv) è una variabile macroscopica e continua. Essendo il risultato del comportamento aggregato di migliaia di utenti, la sua componente aleatoria viene "smussata", facendo emergere una forte periodicità e una chiara autocorrelazione nel tempo. Queste caratteristiche permettono ai modelli regressivi di stimare l'andamento futuro minimizzando efficacemente l'errore, con scarti confinati stabilmente nel range di $0.9 - 1.7\ km/h$.

Al contrario, l'occupazione delle colonnine di ricarica EV è un segnale generato da eventi di singole utenze, in gran parte indipendenti tra loro [3]. Dal punto di vista matematico, la componente aleatoria è dominante e non beneficia della medesima aggregazione macroscopica: si traduce in una serie temporale discreta e impulsiva, assimilabile a una funzione a gradino. I modelli classici faticano a replicare variazioni così repentine e discontinue, portando a un Errore Percentuale (MAPE) compreso tra il 9% e il 24%.

La presenza documentata di una correlazione positiva (confermata dallo *scatterplot*), l'evidenza della derivata temporale (*time-lag*) e la validazione statistica tramite il Test di Granger, dimostrano matematicamente che **il traffico stradale agisce come un predittore indiretto per la rete di ricarica in alcuni contesti urbani**.

In un ecosistema Smart City, questa evidenza abilita un radicale ribaltamento del paradigma ITS: anziché sprecare risorse computazionali per addestrare modelli non lineari sull'imprevedibile disponibilità dei singoli stalli, un sistema centralizzato può concentrare le proprie reti neurali esclusivamente sulla previsione affidabile dei flussi veicolari, questo però solo in alcune zone della città. Superata una determinata soglia di congestione prevista per un'area urbana limitrofa, il sistema può inferire con valenza statistica (da 1 a 2 ore di anticipo) un conseguente picco di occupazione delle infrastrutture EV, operando logiche di instradamento preventivo, allocazione dinamica dei prezzi o gestione anticipata del carico sulla rete elettrica locale (V2G / Smart Grid).

Capitolo 6

Conclusioni

Il presente elaborato di tesi si è posto l'obiettivo di sviluppare e validare un'architettura predittiva orientata ai Sistemi di Trasporto Intelligenti (ITS) [18], affrontando in modo congiunto due dinamiche fondamentali della mobilità urbana: l'evoluzione del traffico stradale e la disponibilità dell'infrastruttura di ricarica per veicoli elettrici (EV). L'applicazione di tecniche avanzate di *Machine Learning* e *Deep Learning* a dati reali, estratti dal tessuto urbano di Milano, ha permesso di delineare un quadro applicativo concreto, evidenziando le potenzialità e i limiti intrinseci degli algoritmi attuali di fronte a segnali di diversa natura fisica e statistica.

6.1 Sintesi del lavoro e risultati chiave

L'analisi metodologica condotta nel corso della ricerca ha evidenziato una netta dicotomia prestazionale. L'applicazione dei modelli predittivi al dominio della congestione veicolare (Δv) ha restituito risultati di buona accuratezza: la natura macroscopica, fluida e spiccatamente autocorrelata del traffico ha permesso agli algoritmi di contenere l'errore medio assoluto nel range di $0.9 - 1.7$ km/h.

Al contrario, la traslazione dei medesimi algoritmi sul tasso di occupazione delle colonnine EV ha esibito evidenti limiti strutturali. L'occupazione di una presa di ricarica, dipendendo da singole decisioni umane [3], è soggetta a varianza e si configura come un segnale discreto e impulsivo (assimilabile a una funzione a gradino). I modelli di *forecasting* classico, intrinsecamente progettati per l'estrapolazione di medie continue, si sono dimostrati inadeguati nel prevedere in modo esatto le repentine transizioni di stato della colonnina, registrando scarti (MAPE) compresi tra il 9% e il 24% (valori che in uno scenario reale inficerebbero l'affidabilità del servizio di prenotazione o instradamento).

A fronte di tale difficoltà metodologica nel dominio diretto, la svolta dell'elaborato risiede nell'aver dimostrato e validato la strada dell'inferenza indiretta. L'analisi di correlazione spaziale e il Test di Causalità di Granger hanno certificato una dipendenza causale e temporale (con ritardi di 1-2 ore) tra i flussi veicolari e l'impegno delle colonnine limitrofe. Poiché il traffico è un segnale altamente e accuratamente prevedibile, esso agisce

come un possibile predittore indiretto per stimare lo stato di saturazione futuro della rete di ricarica.

6.2 Contributi originali e valore dell'elaborato

Il valore scientifico e applicativo del presente lavoro non si esaurisce nell'esecuzione dei test algoritmici, ma si fonda su tre contributi originali specifici:

- **Creazione e Ingegnerizzazione di un Dataset Inedito:** In assenza di banche dati pubbliche pronte all'uso contenenti uno storico granulare sulle colonnine EV della città di Milano, il primo traguardo della tesi è consistito nello sviluppo di uno *scraper* automatizzato per l'estrazione continua dei dati. Tale operazione ha richiesto il superamento pratico dei severi vincoli di chiamata (*rate-limiting*) imposti dai provider, reso possibile unicamente attraverso l'orchestrazione parallela di molteplici chiavi API [5], garantendo una raccolta ininterrotta e la costituzione di una solida *ground truth* per l'addestramento.
- **Analisi Comparativa Multidimensionale:** Il protocollo sperimentale non si è limitato a classificare 5 diverse architetture (Random Forest, SVM, Prophet, Neural Prophet e Stacked LSTM) basandosi su una semplice media d'errore. La valutazione, strutturata su cicli di *Sliding Window* dinamica, ha esaminato la **stabilità temporale** dell'errore (tracciata su orizzonti mensili e in relazione ai fine settimana), definendo un criterio di scelta ingegneristico che affianca alla bontà predittiva la robustezza strutturale, e ha misurato i costi computazionali offline e online.
- **Fattibilità Architetturale per le Smart City:** I test condotti sull'impatto della frequenza di aggiornamento hanno dimostrato che il passaggio dallo *step* giornaliero al *nowcasting* a 30 minuti non produce miglioramenti nell'accuratezza, introducendo al contempo colli di bottiglia computazionali insostenibili per modelli complessi. Questo dato fornisce un'indicazione progettuale diretta: i Sistemi di Trasporto Intelligenti basati su tecniche di machine learning devono operare il ricalcolo dei pesi in modalità *batch* notturna per risultare realmente scalabili su reti urbane estese, garantendo contestualmente latenze di inferenza in produzione misurabili in frazioni di millisecondo.

6.3 Limiti del lavoro

Come ogni ricerca sperimentale basata su acquisizioni empiriche, lo studio presenta alcune limitazioni, di cui si ha piena consapevolezza metodologica:

- **Orizzonte Temporale del Dataset EV:** La raccolta dei dati sull'occupazione delle colonnine, durata approssimativamente cinque mesi e centrata sul periodo primaverile ed estivo, è risultata sufficiente per validare le metriche di breve e medio

termine. Tuttavia, la presenza fisiologica del calo dei flussi di agosto, unita all'assenza di dati invernali e autunnali, ha impedito ai modelli di internalizzare i pattern legati alle stagionalità su scala annuale (limite non presente nel dataset del traffico, monitorato per oltre 18 mesi).

- **Natura Impulsiva e Quantizzazione del Segnale EV:** Come ampiamente discusso, la discretizzazione estrema delle serie EV (spesso quantizzate a step del 50% per colonnine con sole due prese) rende gli algoritmi regressivi tradizionali strutturalmente svantaggiati, introducendo un tetto massimo di precisione intrinsecamente insuperabile se ci si affida esclusivamente all'estrapolazione lineare della media mobile passata.

6.4 Sviluppi futuri

Le evidenze emerse dal lavoro aprono la strada a diverse linee di ricerca e integrazioni industriali:

- **Integrazione dei Large Time Series Models:** Per affrontare in modo più diretto le criticità riscontrate sui segnali impulsivi e ad alta varianza, la naturale evoluzione metodologica coincide con l'adozione dei cosiddetti *Large Time Series Models* (LTSM). Sebbene non implementati nel presente elaborato a causa delle elevate risorse hardware richieste per l'addestramento, i *Foundation Models* sviluppati su architetture Transformer (come *TimesFM* di Google [14] o *Chronos* di Amazon [15]) rappresentano l'attuale stato dell'arte. La loro capacità di operare inferenze *zero-shot* analizzando simultaneamente numerose variabili di contesto costituisce la soluzione tecnologica più promettente per le future evoluzioni dei Sistemi di Trasporto Intelligenti.

- **Estensione Temporale del Dataset EV:** Come rilevato nei limiti dello studio, la raccolta dati sul traffico copre già un solido orizzonte di 18 mesi (da marzo 2025 a settembre 2026). L'estensione della raccolta dati relativa all'occupazione delle colonnine EV (attualmente limitata a circa cinque mesi) a un intero anno solare e oltre, permetterebbe di allineare storicamente i due domini. Questo ampliamento consentirebbe ai modelli di apprendere in modo strutturale l'impatto delle stagionalità a lungo termine e delle singole festività, catturando macro-pattern comportamentali legati a specifici periodi dell'anno (come le dinamiche invernali) e affinando ulteriormente la capacità del traffico di agire come predittore per le ricariche.

Bibliografia

[1] Julio A. Sanguesa, Vicente Torres-Sanz, Piedad Garrido, Francisco J. Martinez, and Johann M. Marquez-Barja. A review on electric vehicles: Technologies and challenges. *Smart Cities*, 4(1):372–404, 2021.

[2] Thomas Franke, Isabel Neumann, Florian Bühler, Peter Cocron, and Josef F. Krems. Experiencing range in an electric vehicle: Understanding psychological barriers. *Applied Psychology*, 61(3):368–391, 2012.

[3] Thomas Franke and Josef F. Krems. Understanding charging behaviour of electric vehicle users. *Transportation Research Part F: Traffic Psychology and Behaviour*, 21:75–89, 2013.

[4] Mostafa Majidpour, Hamidreza Nazaripouya, Peter Chu, Hemanshu R. Pota, and Rajit Gadh. Fast prediction for smart grid concurrent charging of electric vehicles. *IEEE Transactions on Smart Grid*, 6(3):1555–1562, 2015.

[5] TomTom International BV. *TomTom Traffic and Routing APIs Documentation*. TomTom Developer Portal, 2024. `https://developer.tomtom.com/`.

[6] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[7] Rob J. Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018. Disponibile online gratuitamente: `https://otexts.com/fpp3/`.

[8] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[9] Sean J. Taylor and Benjamin Letham. Forecasting at scale. *PeerJ Preprints*, 5:e3190v2, 2017.

[10] Oskar Triebe, Nikolay Laptev, and Ram Rajagopal. Neuralprophet: Explainable forecasting at scale. *arXiv preprint arXiv:2111.15397*, 2021.

[11] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[12] Harris Drucker, Christopher J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems*, pages 155–161, 1997.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[14] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

[15] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

[16] Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4):1–21, 2012.

[17] Joao Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):1–37, 2014.

[18] George Dimitrakopoulos and Panagiotis Demestichas. Intelligent transportation systems. *IEEE Vehicular Technology Magazine*, 5(1):77–84, 2010.

[19] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[20] Karl Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242, 1895.

[21] Charles Spearman. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101, 1904.

[22] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, 37(3):424–438, 1969.