POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Analisi di un sistema di carsharing tramite algoritmi di predizione

Relatori
prof. Paolo Garza
prof. Luca Cagliero
prof.ssa Silvia Chiusano

Candidato
Luca Faccio

Anno accademico 2018-2019

Indice

Capitolo 1

Introduzione

Il concetto di carsharing nasce dall'applicazione dell'idea di sharing economy al trasporto privato. Negli ultimi anni, in molte città del mondo, si sono presentati sul mercato dei trasporti diversi operatori fornitori di questo servizio, proponendolo come un'alternativa od un'integrazione alle classiche modalità di spostamento.
Il sistema del carsharing prevede la condivisione di una flotta di autoveicoli tra più utenti registrati al servizio, i quali possono noleggiare i mezzi generalmente attraverso l'utilizzo di un'applicazione per smartphone.

I sistemi di carsharing si possono distinguere in due tipologie principali: station-based e free-floating. Di queste due, la tipologia free-floating risulta la più interessante dal punto di vista dell'utente finale, dato che puo' iniziare e terminare il noleggio dell'autoveicolo in un qualunque punto di una zona piuttosto estesa, definita a priori dall'operatore, che coincide comunemente con la zona urbana di una città. Questa libertà genera però per l'operatore una complessità nella gestione della flotta, puo' infatti accadere che un utente termini il noleggio in un'area dove è presente una scarsa domanda di autoveicoli, di conseguenza è probabile che quel veicolo rimarrà inutilizzato per un lungo periodo di tempo, riducendo l'efficienza del sistema. Per contrastare questo problema, gli operatori assumono dei dipendenti che hanno come compito lo spostamento delle automobili da zone caratterizzate da una bassa richiesta a zone caratterizzate da una maggiore richiesta, questa operazione viene definita rilocazione. La rilocazione, per essere efficiente, deve essere eseguita conoscendo i luoghi dove in futuro ci sarà richiesta di autoveicoli.

L'infrastruttura informatica, che permette la gestione ed il funzionamento del carsharing, in genere prevede l'esposizione di un'API web, alla quale l'utente si collega utilizzando un'applicazione installata sul proprio smartphone. Attraverso questa applicazione l'utente puo' visualizzare le autovetture disponibili al noleggio in tempo reale, puo' prenotarle per un certo periodo di tempo limitato (variabile a seconda del gestore) e, quando è nei pressi del veicolo, puo' iniziare il noleggio.

Questa interfaccia web è una grande fonte di dati che, se manipolati opportunamente con tecniche di datamining, possono essere usati per caratterizzare e predirre l'utilizzo del servizio.

Le grandi quantità di dati conservate nelle infrastrutture informatiche, che caratterizzano organizzazioni e sistemi dei tipi più disparati, hanno portato negli ultimi anni alla realizzazione e allo sviluppo di algoritmi che permettano di estrarre delle informazioni utili dalla loro analisi. Questo processo di estrazione è comunemente chiamato datamining e sta generando un crescente interesse in molti ambiti, per esempio nel campo medico, dove ha già permesso di effettuare scoperte molto importanti per il progresso della scienza, o anche in campi meno scientifici come quello politico o finanziario, celebre è l'esempio di Obama alle elezioni presidenziali degli Stati Uniti d'America. Uno degli utilizzi del datamining è la creazione di modelli predittivi che, se sufficientemente accurati, possono essere utilizzati per guidare le decisioni da intraprendere al'interno di un'attività di business.

Il lavoro descritto in questa tesi pone come obbiettivo la predizione della disposizione degli autoveicoli in un sistema di carsharing, per fornire un'indicazione utile a massimizzare l'efficienza della rilocazione, mediante l'utilizzo di tecniche di datamining. Il problema di predizione di un valore numerico continuo viene definito nell'ambito dell'analisi dei dati 'regressione'.
Per risolvere questo problema, sono stati confrontati diversi algoritmi predittivi, così da individuarne il migliore, applicando variazioni al modello e analizzando i cambiamenti nelle predizioni. Gli algoritmi che sono stati considerati sono contenuti nella libreria scikit-learn, sviluppata in linguaggio Python, e sono: Support Vector Machines (con differenti kernel), Linear Regression, Gradient Boosting Regressor Trees, Lasso, Random Forest Regressor.
Il caso di studio preso in esame riguarda il servizio di carsharing, erogato dall'operatore car2go, nella città statunitense di Portland. Il dataset utilizzato descrive l'utilizzo del servizio per un periodo temporale lungo circa un anno e mezzo, dal mese di giugno 2012 al mese di dicembre 2013.
La prima fase del lavoro è stata la realizzazione di un modulo per il preprocessing del dataset di input e la preparazione dei dati alla successiva elaborazione. In seguito è stata eseguita un'analisi per osservare i pattern d'utilizzo del servizio e individuare i momenti della giornata e le zone della città più critici. Parallelamente, con lo stesso scopo, è stato ricercato un dataset contenente le informazioni sui 'point-of-interest' presenti nella città di Portland, dalla cui elaborazione sono state create delle mappe che permettono una rapida visualizzazione dell'informazione.
L'ipotesi che le condizioni meteorologiche potessero influenzare l'utilizzo del carsharing ha portato poi alla ricerca di un'ulteriore dataset contenente queste informazioni, il quale dopo un'opportuna elaborazione è stato unito al dataset principale.
Infine il dataset ottenuto e sue variazioni sono stati utilizzati per generare modelli

predittivi con gli algoritmi citati in precedenza e ne sono state comparate le performance.

Questo documento è organizzato nei seguenti capitoli:

- **Capitolo 2: Stato dell'arte**

  Descrive la letteratura presente riguardo al tema del carsharing e di altri sistemi di condivisione di mezzi di trasporto.

- **Capitolo 3: Realizzazione del framework**

  Fornisce il contesto del problema trattato e la descrizione della soluzione proposta e dei linguaggi e programmi utilizzati per realizzarla.

- **Capitolo 4: Analisi spaziotemporale**

  Illustra le osservazioni scaturite dall'analisi dei dati d'utilizzo del sistema di carsharing di Portland.

- **Capitolo 5: Valutazione dei modelli predittivi**

  Illustra gli esperimenti eseguiti con i modelli predittivi, valutandone le performance.

- **Capitolo 6: Conclusioni e sviluppi futuri**

  Riepiloga i risultati ottenuti dagli esperimenti e indica le possibili estensioni del lavoro.

Capitolo 2

Stato dell'arte

Il paradigma della sharing-economy ha dato luce al concetto di sharing-mobility, la strategia di trasporto che prevede l'uso di un veicolo condiviso allo scopo di soddisfare una necessità di spostamento [1]. Il campo dei trasporti sta così volgendo lo sguardo verso l'idea di 'Mobility as a Service', trasformandosi da un sistema incentrato sulla proprietà personale dei veicoli ad uno in cui l'attenzione è spostata sull'effettiva necessità di spostamento delle persone. Questo approccio fa' affidamento alla varietà di vettori di trasporto disponibili al giorno d'oggi e alla semplicità con cui se ne possono consultare le disponibilità, realizzata grazie alle infrastrutture informatiche sempre più veloci ed efficienti [2].
Il servizio di carsharing, come anche quello di bike-sharing, concorrono alla realizzazione dell'idea di 'Mobility as a Service'[3].

2.1 Carsharing, caratteristiche e funzionamento

Il carsharing prevede la condivisione di automobili tra più utenti aderenti al servizio. Questa condivisione è gestita da un operatore che mette a disposizione una flotta di automobili, le quali vengono noleggiate, generalmente per brevi periodi di tempo, dagli utenti.
Il noleggio può venire implementato in due modalità diverse, SB (station-based) o FF (free-floating), o come una combinazione di questi due.
Nella modalità SB l'utente è obbligato a iniziare e terminare il viaggio in un'area definita, come dei parcheggi appositi, che hanno un'estensione geografica ed una capienza limitati. Questa modalità si divide ulteriormente in due sottocategorie, 'one way trip' e 'round trip', la prima permette di terminare il noleggio in una stazione diversa da quella di prelievo del veicolo, mentre la seconda impone che il termine del noleggio avvenga nella stessa stazione in cui era stato iniziato.
La modalità FF invece permette all'utente di muoversi liberamente, iniziando e

terminando il viaggio in punti arbitrari interni all'area operativa definita dall'operatore, che generalmente coincide con l'area urbana delle città. Sono previste inoltre delle piccole zone esterne all'area urbana, chiamate 'zone satellite'. Queste piccole aree sono punti strategici di interesse, in cui è localizzata un'alta richiesta di mezzi, che però sono distaccate dai confini della città. Un ottimo esempio sono gli aereoporti, che spesso sono situati molto lontano dalla città, ma sono punti cruciali e devono essere raggiunti da ogni mezzo di trasporto. Altri casi di 'zone satellite' possono essere aree commerciali con un'alta densità di aziende o campus universitari distaccati dal centro della città[4].
L'utente del servizio di carsharing in genere è tenuto a pagare una quota di iscrizione per accedere al servizio, mentre il costo del singolo viaggio è definito da una combinazione dei seguenti parametri: tipo di veicolo, chilometri percorsi, tempo impiegato, zone speciali raggiunte (per esempio aereoporti) [5]. Il prezzo nei servizi FF è principalmente definito come prezzo al minuto. In questo prezzo il servizio include una quota per i costi variabili, come il carburante, e una quota per i costi fissi di manutenzione, ri-bilanciamento, assicurazione e parcheggio. Il prezzo nei sistemi SB invece è frazionato sull'ora, con dei sovrapprezzi applicati al superamento di una soglia di chilometri percorsi e il tempo minimo di noleggio è di un'ora. Tutti e due i sistemi applicano prezzi differenti in base al modello di autoveicolo scelto [6].
Con l'utilizzo del carsharing l'utente può beneficiare di quasi tutti i privilegi di un'auto privata senza dover affrontare gli oneri legati al possesso di un veicolo personale [7].

2.2 Effetti del carsharing e target di utilizzo

L'obbiettivo principale del carsharing è di ridurre le auto di proprietà [12]. La riduzione dei veicoli di proprietà crea impatti positivi come l'uso più efficiente delle strade e delle infrastrutture viarie e il risparmio di denaro per gli utenti [7]. Per uso più efficiente delle infrastrutture viarie si intende un minor numero di congestionamenti del traffico e più disponibilità di parcheggi, con un conseguente miglior utilizzo dello spazio urbano, si ha inoltre la riduzione delle emissioni generate dai veicoli. Il carsharing è riconosciuto come un concetto di mobilità che può avere un impatto positivo sulla riduzione di emissioni inquinanti e sulla riduzione del traffico cittadino [12] [14], è quindi un valido sistema da prendere in considerazione per l'evoluzione delle smart-cities.
Gli utenti del carsharing sono generalmente giovani persone di sesso maschile, appartenenti alla fascia di età 25-40 anni, con un alto livello di istruzione, che hanno un reddito medio-basso a causa della giovane età e vivono in piccoli nuclei famigliari senza figli [7] [13].
Riguardo all'utilizzo del carsharing, nel paper [10] gli autori lo propongono come

un'integrazione all'uso del veicolo privato per ridurre l'inquinamento. Generalmente una persona acquista un veicolo pensando a soddisfare tutte le necessità che dovrà affrontare, anche quelle più rare, loro invece ipotizzano uno scenario in cui l'utente acquisti un veicolo privato più essenziale e si avvalga del carsharing per quelle sporadiche occasioni in cui si presenta una necessità specifica, per esempio una capacità di carico maggiore. Dividono le caratteristiche del veicolo che vengono utilizzate su base quotidiana da quelle occasionali e valutano se le attuali compagnie di carsharing riescono a fornire queste caratteristiche occasionali con un rapporto spese/benefici apprezzabile dall'utente. Concludono comprendendo che attualmente le compagnie di carsharing non valutano questa idea di funzionamento, ma si concentrano sul fornire il veicolo a utenti che ne sono totalmente sprovvisti.
In [17] vengono analizzati due sistemi di carsharing di Monaco di Baviera dal punto di vista temporale e spaziale, considerando l'influenza di fattori esterni come clima e dati socio-demografici. Vengono notati dei picchi di utilizzo del servizio nella fascia oraria 8-11 del mattino e nella fascia 5-8 pomeridiana. Osservano una sostanziale differenza di pattern di utilizzo tra giorni feriali e finesettimanali ed un'alta concentrazione di partenze ed arrivi intorno alle università ed al centro cittadino. Si accorgono anche che temperatura e pioggia presentano una bassa correlazione con il numero di noleggi, mentre esiste invece una correlazione con la presenza di persone giovani tra gli abitanti della zona.
Sull'analisi dei modi di utilizzo del carsharing si concentra invece il lavoro [11] in cui osservano i dati degli spostamenti dei veicoli di 3 compagnie di FFCS in 22 paesi. Confermano le osservazioni fatte da altri studi: l'uso maggiore del servizio è concentrato durante il mattino nell'orario in cui la maggior parte delle persone si reca a lavoro e alla sera quando ritornano a casa. Notano anche una sostanziale quantità di viaggi 'round-trip' intorno all'ora di pranzo. Un'altra loro scoperta è il fatto che in genere i punti di partenza di un viaggio sono lontani più di 500 metri da una stazione della metropolitana o da un treno regionale, quindi affermano che il FFCS viene usato come complemento e non come alternativa ai mezzi pubblici.
Lo studio [18], che si occupa di analizzare un servizio di carsharing di Basilea, rileva che l'uso del servizio aumenta con l'aumentare della densità di popolazione e, a differenza di altri studi, rileva una correlazione negativa col numero di luoghi di lavoro nella zona. Osservano anche che le persone son disposte a camminare di più per raggiungere un'auto del carsharing piuttosto che i mezzi pubblici e che il FFCS viene usato particolarmente per sopperire alle mancanze di trasporto pubblico. In [**?**] studiano un sistema di carsharing di Montreal e rilevano un'influenza da parte delle stagioni dell'anno sul numero di noleggi, in particolare notano un maggior uso del servizio durante la stagione estiva.

2.3 Criticità del servizio

La libertà, concessa dalla modalità 'free-floating', di lasciare il veicolo in un qualunque punto della zona operativa, origina uno dei maggiori problemi nella gestione della flotta: il problema del ribilanciamento. Nell'area operativa del sistema FF, durante i vari periodi della giornata, si originano delle sottozone chiamate hot-spots e cold-spots. Gli hot-spots sono delle aree con un'alta concentrazione di partenze e arrivi, dove l'intensità del flusso di veicoli è alta. Al contrario, i cold-spots sono aree dove il numero di arrivi è maggiore del numero di partenze e si origina quindi un 'ristagno' di veicoli. Questa situazione obbliga l'operatore fornitore del servizio ad attuare un'operazione di ribilanciamento, spostando parte della flotta e riposizionandola in aree dove ci sarà una futura concentrazione di richieste di veicoli [8]. Una strategia alternativa attuabile dall'operatore è rappresentata dal creare incentivi per l'utente, come sconti sulle tariffe, applicati per l'utilizzo dei veicoli presenti in un cold-spot. La stessa tecnica può essere attuata per incentivare le operazioni di rifornimento delle auto con bassa autonomia residua (carburante o batterie nel caso di veicoli elettrici). Il problema del ribilanciamento non è presente nei sistemi SB 'round trip', perchè ogni stazione ha la sua capacità allocata a priori dall'operatore. Infatti, la modalità di carsharing SB si concentra su aree relativamente piccole e il bilancio dei veicoli è tenuto fisso grazie al vincolo di rilasciare l'automobile nello stesso punto dove era stato iniziato il viaggio. In ogni caso tutti e due i sistemi devono tener conto dei compiti di manutenzione e di rifornimento dei veicoli.
Entrambe le modalità presentano vantaggi e svantaggi per utenti e operatori. Per gli utenti i quali richiedono un alto grado di libertà, il FFCS (free-floating carsharing) soddisfa questa richiesta, a fronte di un costo maggiore del servizio in confronto alla tariffa oraria proposta dalla modalità SBCS (station-based carsharing), inoltre il parcheggio non è garantito e questo può causare ulteriore tempo speso nel veicolo (e di conseguenza costi più alti). Per gli operatori, il FFCS è un modello di business più allettante, ma ha degli alti costi di ribilanciamento, dato che ogni auto necessita potenzialmente di una persona che la riposizioni a seconda della domanda degli utenti. Il SBCS è invece più economico da implementare, ma pone delle limitazioni all'utente. Per viaggi che hanno come scopo il raggiungimento del luogo di lavoro il FFCS rappresenta il sistema più funzionale invece, per viaggi durante il tempo libero, il SBCS può offrire una soluzione più economica [9].

Alcune ricerche svolte finora per migliorare i sistemi di carsharing si focalizzano su 2 macro-categorie: ottimizzazione e predizione della domanda.

- Ottimizzazione

  Nel lavoro [15] viene presentato un modello matematico per riconoscere il miglior numero di stazioni e la loro miglior locazione a Nizza, in Francia. I punti

chiave che influenzano la presenza di stazioni sono: alto reddito e alta educazione dei residenti, presenza di centri commerciali, presenza di alberghi e alta densità di popolazione.
In [16] affrontano il problema dei costi elevati per eseguire la rilocazione. Propongono delle tariffe con prezzi dinamici, che favoriscano una rilocazione dei veicoli eseguita dagli utenti, ottenuta incentivando economicamente gli utenti a svolgere i viaggi che invece dovrebbe eseguire un'operaio dedicato, dipendente dell'azienda erogatrice del servizio.

- Predizione

  Un tentativo di predizione a breve termine dei noleggi viene effettuato in [20], dove viene studiato il caso della compagnia DriveNow a Berlino. L'area operativa viene divisa in zone con differenti ZIP code, per ognuna delle quali viene calcolata una predizione. Utilizzano due metodi per l'analisi delle serie temporali: un modello 'seasonal ARIMA' e l' 'exponential smoothing with Holt-Winters-Filter'. L'HWF dimostra di avere tempi computazionali e risultati migliori, performando al meglio con l'utilizzo di un training set di 3 mesi. L'uso di una rete neurale evolutiva viene sperimentato invece in [22], dove viene riscontrata una performance migliore in confronto al classico approccio di predizione basato su time series.

Capitolo 3

Realizzazione del framework

3.1 Caratteristiche dei dataset

Il dataset principale utilizzato in questa tesi, reperito da [23], si riferisce al servizio di carsharing, implementato dall'operatore car2go a Portland, per un periodo temporale lungo un anno e mezzo, da giugno 2012 a dicembre 2013, con una flotta composta da un totale di 316 automobili. Contiene i dati relativi alla posizione di ogni automobile libera in un dato momento, con una granularità temporale di 15 minuti. Il dataset scaricato è stato realizzato attraverso l'utilizzo di un web crawler, che ha collezionato i dati di disponibilità dei veicoli contattando periodicamente l'API del servizio ospitata sui server di car2go, ottenendo così più di 10 milioni di righe di dati.
Come si può vedere nell'immagine 3.1, ogni riga del dataset contiene i seguenti campi:

- *date*: il timestamp durante il quale è stata rilevata l'informazione contenuta nella riga
- *license*: il codice identificativo del veicolo al quale si riferisce la riga
- *latitude*: la latitudine della posizione del veicolo
- *longitude*: la longitudine della posizione del veicolo
- *fuel*: il livello del carburante espresso in percentuale
- *interior*: valore booleano che indica lo stato degli interni
- *exterior*: valore booleano che indica lo stato della carrozzeria

Un altro dataset impiegato in questo lavoro è stato ottenuto da [24], è composto da 6 file contenenti distintamente le informazioni su umidità, pressione atmosferica, temperatura, direzione del vento, velocità del vento e descrizione qualitativa del

date	license	latitude	longitude	fuel	interior	exterior
2013-04-01 00:00:08	340FRH	45.54935	-122.5807	100	true	true
2013-04-01 00:00:09	335FRH	45.50137	-122.67981	100	true	true
2013-04-01 00:00:09	334FRH	45.52929	-122.6869	45	true	true
2013-04-01 00:00:09	681FZK	45.50052	-122.62328	84	true	true
2013-04-01 00:00:09	332FRH	45.5223	-122.62588	45	false	true
2013-04-01 00:00:09	331FRH	45.57185	-122.63175	66	true	true
2013-04-01 00:00:09	328FRH	45.58012	-122.71857	90	true	true
2013-04-01 00:00:09	166FRH	45.50742	-122.6548	63	true	true
2013-04-01 00:00:09	173FRH	45.48232	-122.64762	100	true	true

Figura 3.1: Esempio dataset delle posizioni delle automobili

meteo per 30 città americane e canadesi. La granularità temporale con cui sono raccolte le osservazioni è di un'ora e il dataset copre 5 anni, dal 2012 a 2017.
Ogni riga di ogni file contiene un campo datetime che indica il momento di rilevazione dei dati ed un campo per ogni città dove viene indicato il valore del dato caratteristico di quel file per quella città, come si può osservare nell'immagine 3.2.
Per elaborare questi file è stato realizzato uno script in linguaggio Python. Da ogni file corrispondente ad una condizione meteorologica sono state estratte solamente le colonne relative alla città di Portland e sono state combinate insieme in un unico file. Il file corrispondente alla descrizione qualitativa del meteo è stato scartato.

datetime	Vancouver	Portland	San Francisco	Seattle	Los Angeles	San Diego
2012-10-01 13:00:00	76	81	88	81	88	82
2012-10-01 14:00:00	76	80	87	80	88	81
2012-10-01 15:00:00	76	80	86	80	88	81
2012-10-01 16:00:00	77	80	85	79	88	81
2012-10-01 17:00:00	78	79	84	79	88	80
2012-10-01 18:00:00	78	79	83	78	88	80
2012-10-01 19:00:00	79	78	82	77	88	80
2012-10-01 20:00:00	79	78	81	77	88	79
2012-10-01 21:00:00	80	77	80	76	88	79

Figura 3.2: Esempio file humidity.csv

Il terzo ed ultimo dataset riguarda i 'point of interest', di seguito definiti POI, presenti nella zona coperta dal servizio di carsharing di Portland. Questo dataset è stato ottenuto dal sito web [25], dove è stata definita l'area operativa del carsharing di Portland e il formato di file '.pbf' per estrarre dal file Planet.osm tutti i dati interni all'area selezionata. Il file Planet.osm è il file, aggiornato settimanalmente, che contiene tutti i dati che caratterizzano la mappa mondiale di OpenStreetMap [26]: 'nodes', 'ways', 'relation'. Grazie all'applicazione Java per linea di comando 'Osmosis', il file scaricato è stato manipolato, estraendone solamente i 'nodes' in

un file di formato ‘.osm‘ (un formato codificato in xml).

3.2 Linguaggi e programmi utilizzati

Il software di questo lavoro di tesi è stato scritto in due linguaggi di programmazione, inizialmente Java e successivamente Python.
Java è un linguaggio di programmazione ad alto livello, orientato agli oggetti e a tipizzazione statica, che si appoggia sull'omonima piattaforma software, specificamente progettato per essere il più possibile indipendente dalla piattaforma hardware di esecuzione (tramite compilazione in bytecode prima e interpretazione poi da parte di una JVM) [29].
Il codice Java è stato scritto all'interno dell'ambiente di sviluppo integrato *Eclipse*.
La parte di join dei dati e quella di predizione si è scelto di svilupparle in Python.
Python è un linguaggio multi-paradigma che ha tra i principali obiettivi dinamicità, semplicità e flessibilità. Supporta il paradigma object oriented, la programmazione strutturata e molte caratteristiche di programmazione funzionale e riflessione. Le caratteristiche più immediatamente riconoscibili di Python sono le variabili non tipizzate e l'uso dell'indentazione per la definizione delle specifiche. Altre caratteristiche distintive sono l'overloading di operatori e funzioni tramite delegation e la presenza di un ricco assortimento di tipi e funzioni di base e librerie standard [30].
Il codice in questo caso è stato prodotto utilizzando l'IDE *PyCharm*.
Di questo ricco assortimento, in questo lavoro sono state sfruttate le librerie ‘pandas‘, ‘scikit-learn‘ e ‘folium‘.
Pandas è una diffusa libreria per Python che permette di implementare efficacemente programmi per la manipolazione e l'analisi dei dati. Mette a disposizione l'oggetto dataframe, che è strutturato concettualmente come una classica tabella, evitando la gestione dei dati attraverso liste o dizionari. Su di esso si possono operare rapide operazioni di modifica dei dati, memorizzazione su file e lettura da file.
Scikit-learn [31] è una libreria per Python che implementa svariati algoritmi di apprendimento supervisionato e non, in particolare algoritmi di classificazione, regressione, clustering e preprocessing. Si appoggia alle librerie ‘NumPy‘, ‘pandas‘ e ‘Matplotlib‘.
Gli algoritmi di questa libreria che sono stati utilizzati in questa tesi sono:

- *SVR*, è l'implementazione delle ‘Support Vector Machines‘ per il problema della regressione, in questo lavoro è stato testato con 3 differenti kernel: lineare, gaussiano (rbf) e sigmoidale;
- *LinearRegression*, è un modello lineare che utilizza il metodo dei minimi quadrati;
- *Lasso*, è anch'esso un modello lineare che utilizza il metodo dei minimi quadrati ma con l'aggiunta di un termine di regolarizzazione;

- *GBRT*, Gradient Boosting Regression Trees fa parte dei metodi di 'ensemble', è basato sul modello dell'albero decisionale;

- *RF*, Random Forest Regressor è anch'esso un metodo di 'ensemble' basato sul modello dell'albero decisionale.

Folium è una libreria che permette la creazione di mappe interattive integrando in Python la libreria javascript 'Leaflet', le mappe create sono quindi visualizzabili e gestibili mediante un browser. Nonostante alcune funzionalità di 'Folium' siano risultate di difficile gestione, è stato un'efficace strumento per la produzione di rappresentazioni visive, quali heatmap, nel contesto di questo lavoro di tesi.

3.3 Overview del problema e soluzione proposta

Il problema che questa tesi mira a risolvere è il problema della rilocazione degli autoveicoli in un sistema di carsharing. Per effettuare una rilocazione efficiente che soddisfi la richiesta dei clienti, è necessario riposizionare le automobili in punti adiacenti a dove ne verrà fatta richiesta. Predirre la disposizione degli autoveicoli nel breve termine risulta essere un'informazione preziosa per il gestore del servizio, che può usarla per ottimizzare la rilocazione. Ai fini del problema ogni veicolo vale quanto un altro, quindi la predizione riguarda la disposizione dell'intera flotta sull'area del servizio. L'area del servizio è stata modellata dividendola in 200 celle e il framework fornisce una predizione distinta per ognuna delle celle.

Una predizione è caratterizzata dall'istante di tempo nel futuro per cui la si fornisce, definiremo *orizzonte* il numero di intervalli di tempo che intercorrono tra il momento in cui si effettua la predizione e il momento per il quale la si fornisce.

Il problema è stato modellato in due versioni, definite 'assoluta' e 'relativa':

- Nella versione **assoluta** il valore da predirre, e quindi l'occupazione della singola cella, è definita come il numero di veicoli presenti all'interno di una circonferenza di raggio pari a 500 metri, il cui centro coincide con il centro della cella, al tempo $T = t+h$, dove h è l'orizzonte e t è il momento attuale.

- Nella versione **relativa** invece il valore da predirre è definito come la differenza tra il numero di veicoli presenti all'interno della circonferenza (di raggio pari a 500 metri, il cui centro coincide con il centro della cella) al tempo $T = t+h$ (definito come prima) e la stessa quantità al tempo t.

Il modello elaborato basa le sue predizioni sulle serie storiche dell'occupazione delle celle e, considerata l'ipotesi di lavori precedenti riguardo al fatto che il meteo potrebbe influenzare l'utilizzo del carsharing, sulle condizioni meteorologiche di

Portland. Le features del dataset di input sono state inoltre estese con delle features legate al timestamp delle serie storiche.
Per elaborare una predizione il modello prevede di utilizzare, oltre allo storico dell'occupazione interna alla circonferenza di raggio pari a 500 metri, anche lo storico dell'occupazione interna a delle corone circolari di raggio maggiore, che in questo lavoro sono state definite con raggio pari a 1500 e 2000 metri, dalla figura 3.3 si può comprendere meglio: l'area rossa è quella interna alla circonferenza di raggio 500 metri, mentre le aree arancione e gialla sono le corone circolari definite rispettivamente dai raggi di lunghezza 1500 e 2000 metri.

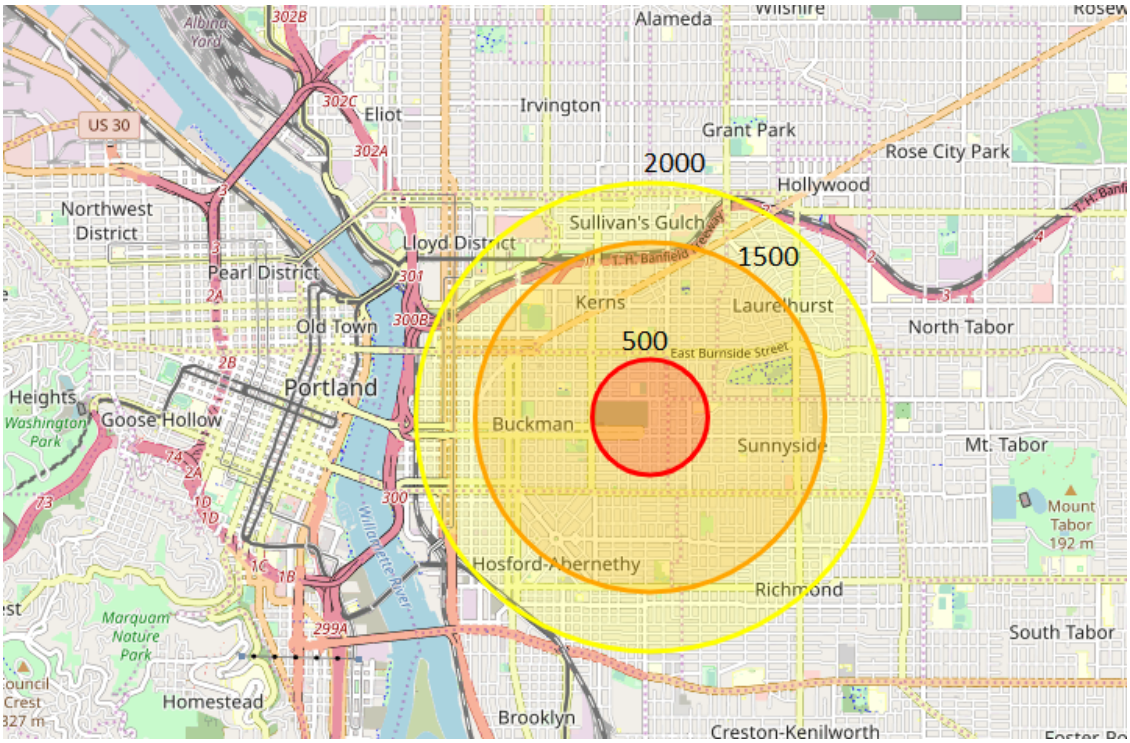


Figura 3.3: Rappresentazione delle aree per cui viene considerata l'occupazione di veicoli

Le stesse considerazioni valgono anche per la versione *relativa*, dove le serie storiche riguardano però il valore della differenza tra l'occupazione interna alle aree al tempo considerato e l'occupazione all'istante precedente.

I modelli precedentemente descritti definiscono così il dataset di input del processo predittivo, il quale può essere configurato sui parametri:

- *Cella:* definisce la cella per la quale si desidera la predizione.
- *Numero di raggi e loro valore:* serve a definire il numero di corone circolari

esterne alla circonferenza per le quali si vuole considerare l'influenza sulla predizione, fissato questo numero ne consegue la definizione dei valori dei raggi.

- *Horizon:* definisce il numero di intervalli di tempo che caratterizzano la predizione, è un valore intero per il quale un'unità corrisponde a 15 minuti (un horizon uguale a 4 corrisponderà ad 1 ora).

Il framework si occupa di valutare la bontà della predizione prodotta dal processo predittivo, questo processo è caratterizzato da svariati parametri che, sommandosi ai precedenti, caratterizzano la predizione nel suo insieme:

- *Window:* definisce il numero di istanti di tempo per cui vengono considerate le serie storiche, è un valore intero. Per esempio, con un valore di window pari a 5, verranno considerate le occupazioni degli ultimi 5 istanti di tempo, che significa considerare lo storico delle occupazioni fino al tempo $T = t - 4$.
- *Meteo:* determina se le features riguardanti le condizioni meteorologiche devono essere incluse nel processo di predizione o no.
- *Feature selection:* determina se deve essere applicato un processo di feature selection al dataset di input o no.
- *Algoritmo:* è l'algoritmo predittivo utilizzato per generare la predizione, scelto tra i seguenti: Support Vector Machine (con differenti kernel), Linear Regression, Gradient Boosting Regressor, Lasso, Random Forest Regressor.

3.4 Sviluppo del framework

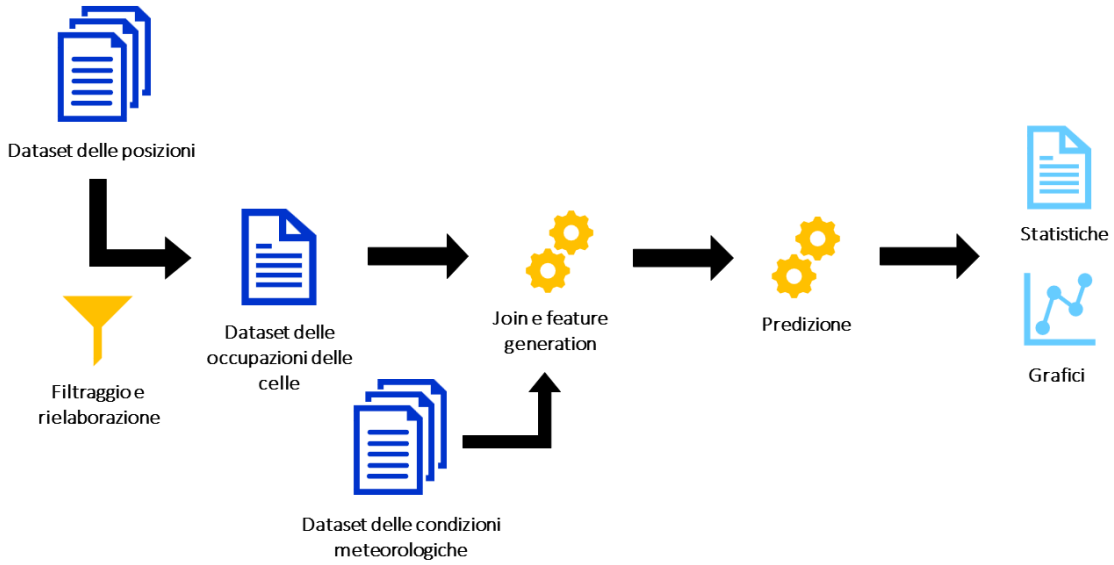
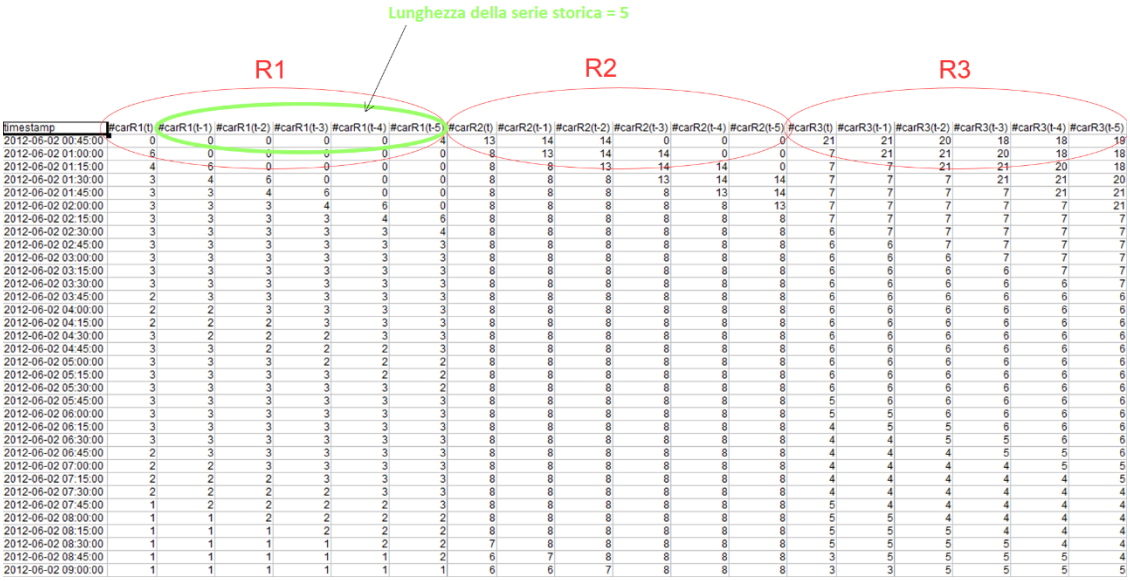


Figura 3.4: Rappresentazione grafica del framework

3.4.1 Elaborazione del dataset delle posizioni

La realizzazione del framework è iniziata con la creazione del programma che processa il dataset delle posizioni, questo programma è stato scritto in linguaggio Java e si occupa di generare il dataset parziale che contiene le serie storiche dell'occupazione delle celle.
Il processamento dei dati prevede la ricostruzione della cronologia delle posizioni di ogni automobile, la quale viene analizzata allo scopo di depurare il dataset da eventuali errori. Non è infatti improbabile che il sistema gps delle automobili generi degli errori nella segnalazione della posizione, inoltre anche il procedimento di acquisizione dati, usato sull'API del servizio per collezionare i dati, non è esente da errori, per esempio le doppie campionature per uno stesso timestamp. La ricostruzione della cronologia delle posizioni per ogni automobile permette quindi di eliminare quelle posizioni che risultano inconsistenti o impossibili da raggiungere in uno scenario reale.
In seguito il programma procede con il conteggio del numero di veicoli per ogni cella per ogni istante di campionamento, ossia per ogni timestamp presente nel dataset di input, e, se son state definite delle corone circolari, al conteggio del numero di veicoli presenti nelle corone circolari.

Infine viene svolta un'operazione di windowing sui dati, così da generare un dataset intermedio per cui in ogni riga, alla quale corrisponde un timestamp, sono presenti le serie storiche parziali delle occupazioni della cella e delle corone circolari. La lunghezza a cui vengono troncate le serie storiche di ogni riga è configurabile, in questo lavoro è sempre stata impostata in modo da avere 20 campioni.
Nella figura 3.5 si può vedere come si presenta il dataset parziale, in questo caso le serie storiche sono state troncate al quinto valore per fini esemplificativi; il numero di raggi è stato impostato a 3, 'R1' è il raggio della circonferenza mentre 'R2' ed 'R3' sono quelli delle corone circolari, il campo timestamp avanza di 15 minuti ad ogni riga.



timestamp	#carR1(t)	#carR1(t-1)	#carR1(t-2)	#carR1(t-3)	#carR1(t-4)	#carR1(t-5)	#carR2(t)	#carR2(t-1)	#carR2(t-2)	#carR2(t-3)	#carR2(t-4)	#carR2(t-5)	#carR3(t)	#carR3(t-1)	#carR3(t-2)	#carR3(t-3)	#carR3(t-4)	#carR3(t-5)
2012-06-02 00:45:00	0	0	0	0	0	4	13	14	14	0	0	0	21	21	20	18	18	19
2012-06-02 01:00:00	6	0	0	0	0	0	8	13	14	14	0	0	7	21	21	20	18	18
2012-06-02 01:15:00	4	6	0	0	0	0	8	8	13	14	14	0	7	7	21	21	20	18
2012-06-02 01:30:00	3	4	6	0	0	0	8	8	8	13	14	14	7	7	7	21	21	20
2012-06-02 01:45:00	3	3	4	6	0	0	8	8	8	8	13	14	7	7	7	7	21	21
2012-06-02 02:00:00	3	3	3	4	6	0	8	8	8	8	8	13	7	7	7	7	7	21
2012-06-02 02:15:00	3	3	3	3	4	6	8	8	8	8	8	8	7	7	7	7	7	7
2012-06-02 02:30:00	3	3	3	3	3	4	8	8	8	8	8	8	6	7	7	7	7	7
2012-06-02 02:45:00	3	3	3	3	3	3	8	8	8	8	8	8	6	6	7	7	7	7
2012-06-02 03:00:00	3	3	3	3	3	3	8	8	8	8	8	8	6	6	6	7	7	7
2012-06-02 03:15:00	3	3	3	3	3	3	8	8	8	8	8	8	6	6	6	6	7	7
2012-06-02 03:30:00	3	3	3	3	3	3	8	8	8	8	8	8	6	6	6	6	6	7
2012-06-02 03:45:00	2	3	3	3	3	3	8	8	8	8	8	8	6	6	6	6	6	6
2012-06-02 04:00:00	2	2	3	3	3	3	8	8	8	8	8	8	6	6	6	6	6	6
2012-06-02 04:15:00	2	2	2	3	3	3	8	8	8	8	8	8	6	6	6	6	6	6
2012-06-02 04:30:00	3	2	2	2	3	3	8	8	8	8	8	8	6	6	6	6	6	6
2012-06-02 04:45:00	3	3	2	2	2	3	8	8	8	8	8	8	6	6	6	6	6	6
2012-06-02 05:00:00	3	3	3	2	2	2	8	8	8	8	8	8	6	6	6	6	6	6
2012-06-02 05:15:00	3	3	3	3	2	2	8	8	8	8	8	8	6	6	6	6	6	6
2012-06-02 05:30:00	3	3	3	3	3	2	8	8	8	8	8	8	6	6	6	6	6	6
2012-06-02 05:45:00	3	3	3	3	3	3	8	8	8	8	8	8	5	6	6	6	6	6
2012-06-02 06:00:00	3	3	3	3	3	3	8	8	8	8	8	8	5	5	6	6	6	6
2012-06-02 06:15:00	3	3	3	3	3	3	8	8	8	8	8	8	4	5	5	6	6	6
2012-06-02 06:30:00	3	3	3	3	3	3	8	8	8	8	8	8	4	4	5	5	6	6
2012-06-02 06:45:00	2	3	3	3	3	3	8	8	8	8	8	8	4	4	4	5	5	6
2012-06-02 07:00:00	2	2	3	3	3	3	8	8	8	8	8	8	4	4	4	4	5	5
2012-06-02 07:15:00	2	2	2	3	3	3	8	8	8	8	8	8	4	4	4	4	4	5
2012-06-02 07:30:00	2	2	2	2	3	3	8	8	8	8	8	8	4	4	4	4	4	4
2012-06-02 07:45:00	1	2	2	2	2	3	8	8	8	8	8	8	5	4	4	4	4	4
2012-06-02 08:00:00	1	1	2	2	2	2	8	8	8	8	8	8	5	5	4	4	4	4
2012-06-02 08:15:00	1	1	1	2	2	2	8	8	8	8	8	8	5	5	5	4	4	4
2012-06-02 08:30:00	1	1	1	1	2	2	7	8	8	8	8	8	5	5	5	5	4	4
2012-06-02 08:45:00	1	1	1	1	1	2	6	7	8	8	8	8	3	5	5	5	5	4
2012-06-02 09:00:00	1	1	1	1	1	1	6	6	7	8	8	8	3	3	5	5	5	5

Figura 3.5: Esempio dataset intermedio

3.4.2 Combinazione dei dataset e generazione di feature temporali

Ottenuto il dataset di occupazione delle celle e quello delle condizioni meteorologiche di Portland, il modulo del framework che è stato implementato successivamente è stato quello per combinare questi dati e per arricchirli. In questo caso il programma è stato scritto in linguaggio Python.
Il primo passo è stato effettuare un'operazione di join tra i due dataset mediante il campo timestamp. Le feature meteorologiche che sono state effettivamente unite al dataset di input sono: 'humidity', 'temperature', 'pressure', 'windspeed'. In questo caso l'accorgimento di maggior importanza ha riguardato il fatto che il join non può essere effettuato direttamente tra le tuple corrispondenti allo stesso

timestamp, questo perchè nel processo predittivo finale l'occupazione al tempo t verrà scelta come label da predirre, se l'operazione di join fosse stata eseguita in questo modo, le features meteorologiche sarebbero state relative al tempo t, ossia relative al futuro rispetto al tempo in cui è elaborata la predizione.
Il join corretto è stato invece eseguito generando una tupla che contiene le informazioni meteorologiche relative al tempo in cui è elaborata la predizione, ossia al tempo $T = t - h$.
In seguito sono state generate delle feature aggiuntive riguardanti il tempo, legate quindi al campo timestamp, nel particolare:

- *Hour:* è la parte estratta dal timestamp che rappresenta l'ora nel formato 'hh'.
- *TimeSlot:* rappresenta la fascia oraria della giornata a cui appartiene la tupla, la giornata è divisa in 8 fasce di ampiezza 3 ore (0-3, 3-6, 6-9, 9-12, 12-15,...).
- *Day:* è la parte estratta dal timestamp che rappresenta il giorno del mese nel formato 'dd'.
- *Weekday:* è un numero che rappresenta il giorno della settimana.
- *Holiday:* è un valore booleano che indica se il giorno a cui appartiene la tupla è un giorno festivo, per generare questa feature è stato utilizzato il modulo 'USFederalHolidayCalendar' della libreria Python 'pandas'.
- *Month:* è la parte estratta dal timestamp che rappresenta il mese nel formato 'MM'.
- *Year:* è la parte estratta dal timestamp che rappresenta l'anno nel formato 'aaaa'.
- *DayOffset:* indica il giorno dell'anno a cui appartiene la tupla.
- *Autumn, Spring, Summer, Winter:* sono dei valori booleani che indicano l'appartenenza della tupla alla stagione.

3.4.3 Predizione e valutazione della sua precisione

L'ultima parte del framework è quella in cui vengono realizzate le predizioni vere e proprie. Il programma, sviluppato in Python, riceve in input il dataset così come è stato costruito nei passi precedenti. Mediante la dichiarazione di svariati parametri si decidono quali predizioni far effettuare al programma, il quale ne valuta le performance.
Il framework effettua le predizioni utilizzando 2 settimane come training set ed 1 giorno come test set (questi parametri sono modificabili nel codice) e valuta la bontà della predizione calcolando il *Mean Absolute Error*, in seguito chiamato MAE, di

tutte le predizioni effettuate per il test set. Questa operazione viene iterata per 30 volte (anche questo parametro è modificabile nel codice) avanzando ogni volta di un giorno nel dataset, si otterranno quindi 30 MAE relativi alle predizioni di 30 giorni del dataset. Anche queste misure a loro volta verranno sintetizzate calcolandone il MAE. I MAE parziali e quello finale vengono memorizzati in file di output per futuri confronti ed elaborazioni.
Questo procedimento viene ripetuto dal framework per ogni combinazione dei parametri che l'utente può definire, alcuni già descritti in precedenza qui 3.3.
Un parametro che non era ancora stato nominato è il giorno dal quale far iniziare il processo di predizione ed è nominato solamente ora perchè prima non ne sarebbe stato compreso lo scopo.
In questa fase dell'elaborazione alcuni parametri sono inalterabili, perchè intrinsechi alla composizione del dataset intermedio di input, in particolare sono: la scelta della versione, l'horizon ed il valore dei raggi (ma è possibile considerarne solo un sottinsieme).
I parametri gestibili invece sono: giorno di inizio della porzione di dataset oggetto del processo, ampiezza della window delle serie storiche(limitata dal valore scelto nel modulo precedente per il windowing), identificativo della cella, algoritmo, esclusione delle feature meteorologiche ed esclusione del processo di feature selection.
Per la visualizzazione e l'analisi delle prestazioni è stata infine integrata una parte che genera dei grafici riassuntivi.

3.5 Programmi accessori

È stato scritto un programma in linguaggio Java per filtrare il file '.osm' e selezionare solamente i nodi che corrispondono ai POI interessanti per l'analisi dell'area operativa del carsharing. I 'point of interest' sono stati classificati nelle seguenti categorie: education, leisure, office, restaurants, shop, sport, supermarket, university. Con un ulteriore processamento, sono stati calcolati il numero di POI per ogni cella distintamente per ogni categoria e oltre alla semplice memorizzazione su un file '.csv' è stata realizzata una mappa, mediante la creazione di uno script in Python che sfrutta la libreria Folium, per facilitarne la comprensione. Questa mappa, visualizzabile attraverso il browser, permette di selezionare la categoria dei poi desiderata e ne mostra la heatmap specifica completa dei conteggi dei poi per cella, come si può veder nelle figure 3.6 3.7.
In un certo momento è stata considerata l'idea di utilizzare i dati relativi ad i POI anche nel processo predittivo, ma due motivi hanno portato ad escludere questa ipotesi. In primo luogo i POI così ricavati non sono esaustivi ed ancor peggio possono essere non attendibili, questo perchè sono stati ricavati dalla fonte OpenStreetMap, che è un progetto aperto che affida il 'mapping' al contributo di volontari non professionisti. Inoltre se i POI fossero usati per caratterizzare una cella (come ipotizzato) non si otterrebbe alcun vantaggio, perchè in ogni caso la predizione viene

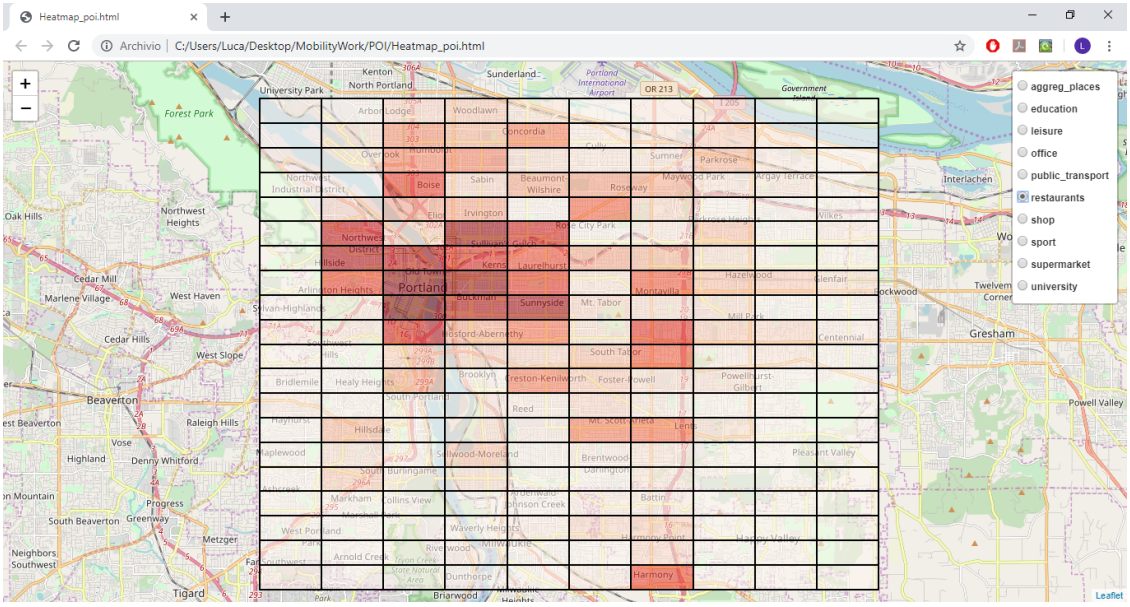


Figura 3.6: Screenshot della heatmap

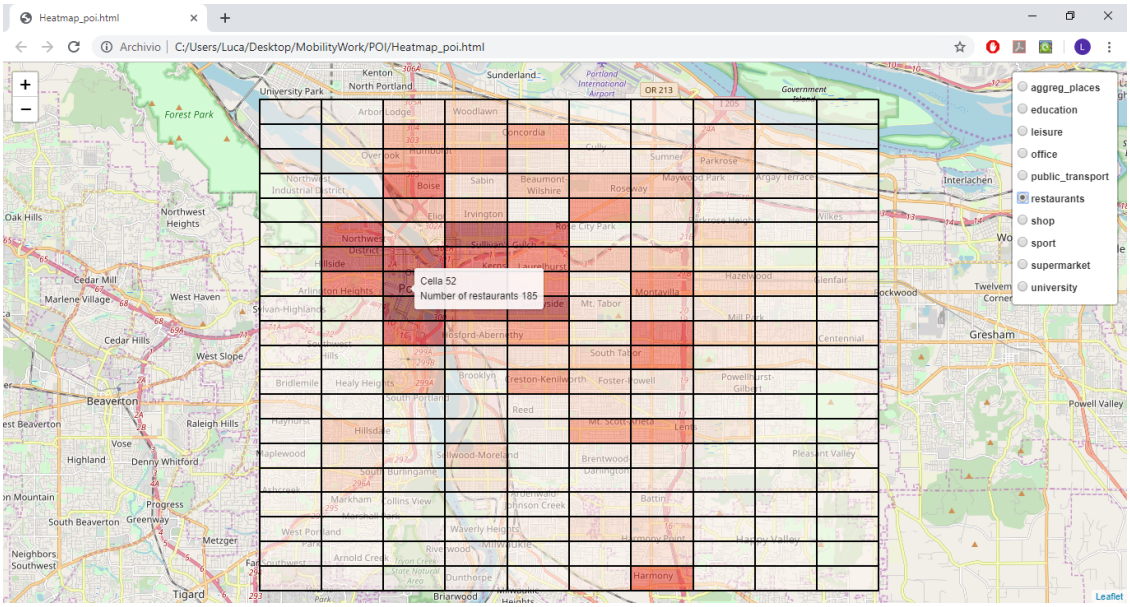


Figura 3.7: Screenshot della heatmap con puntatore posizionato sulla cella

eseguita dal framework distintamente per ogni cella.

Capitolo 4

Analisi spaziotemporale

4.1 Controllo del dataset delle posizioni

La prima analisi effettuata ha riguardato la verifica della consistenza del dataset delle posizioni e la ricerca di errori al suo interno. Le osservazioni sono scaturite prevalentemente durante lo sviluppo della parte di elaborazione del dataset delle posizioni descritta nella sezione 3.4.1, che ha infatti previsto la realizzazione di un meccanismo per depurare il dataset dagli errori.
È stato individuato un buco nei dati corrispondente al periodo che parte dalla data 2012-10-19 e termina alla data 2012-12-23, probabilmente imputabile ad una terminazione indesiderata del processo di acquisizione dati dall'API del servizio.
Sono stati rilevati casi di campionamenti della posizione di un'automobile distanti temporalmente meno di un minuto, che son stati interpretati come doppi campionamenti, questi imputabili sicuramente a qualche malfunzionamento del processo di acquisizione.
Altri errori tipici individuati, probabilmente causati da malfunzionamenti del gps delle automobili, sono coppie di posizioni temporalmente contigue distanti pochi metri l'una dall'altra o così distanti che non sarebbero realizzabili nella realtà.

4.2 Analisi spaziotemporale dell'utilizzo del servizio

In seguito alla generazione del dataset dell'occupazione delle celle ne è stato analizzato l'andamento nel tempo per ogni cella, aggregando i dati su diversi periodi temporali. A questo proposito sono stati realizzati dei grafici per visualizzare l'informazione.
L'analisi è stata svolta considerando solamente la parte di dati relativi all'anno 2013,

questo per evitare che il buco nei dati riferiti al periodo 2012-10-19/2012-12-23 desse origine ad una valutazione non corretta.

4.2.1 Celle inattive

Durante questa analisi si è scoperto che alcune celle dell'area coperta dal servizio non presentano alcuna attività, cioè che al loro interno non sono mai state presenti delle automobili appartenenti al servizio.
Con il proposito di escludere queste celle 'inattive' dall'analisi spaziotemporale è stato eseguito un conteggio del numero totale di istanti in cui un'automobile è disponibile per ogni cella nell'arco dell'intero anno. Questo conteggio ha portato a scoprire che 79 celle non presentano alcuna attività, sono state quindi escluse dall'analisi.
Nella heatmap visibile nell'immagine 4.1 si può osservare la distribuzione dell'attività del servizio, si noti come la parte destra dell'area sia coinvolta da un'attività pressochè nulla.

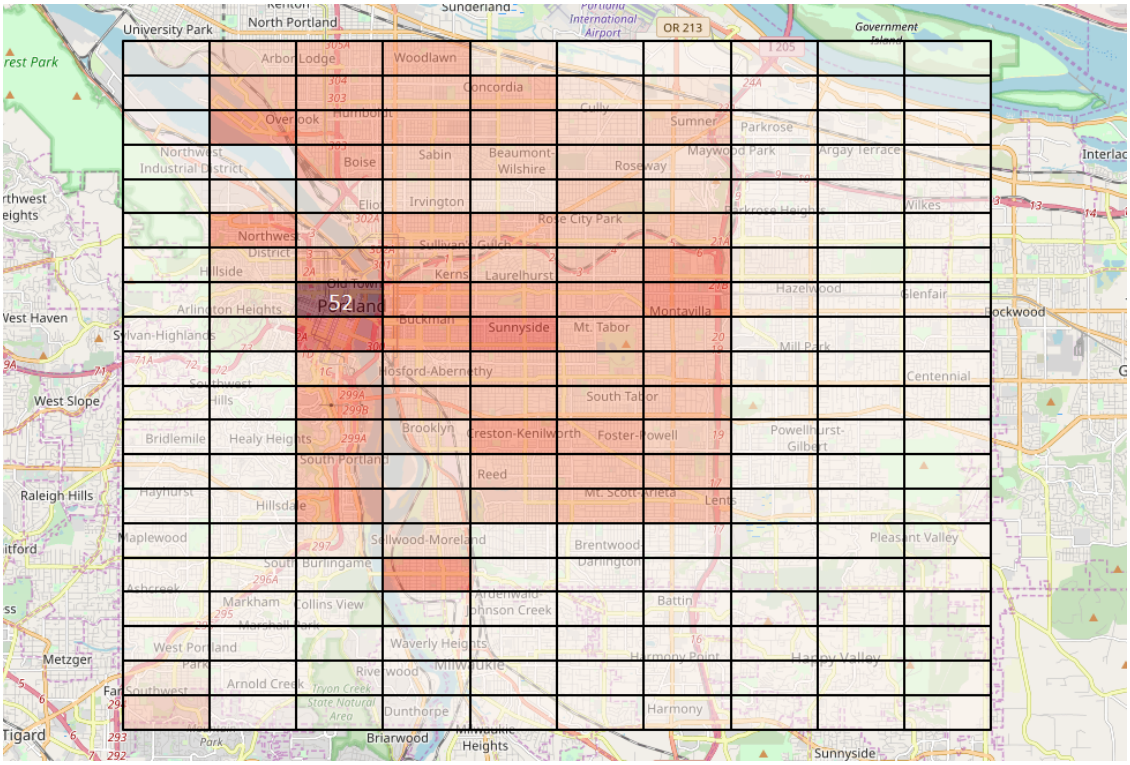


Figura 4.1: Heatmap sul conteggio della disponibilità dei veicoli

È interessante sapere che l'operatore car2go dal 24 agosto 2015 ha deciso di ridurre l'area coperta dal servizio per la città di Portland ed ha escluso proprio quella parte a destra che è stata etichettata come 'inattiva' in questa analisi.

Dalle figure 4.2 e 4.3 si può apprezzare l'entità della riduzione che ha subito l'area operativa; l'azienda ha dichiarato che, basandosi sull'analisi dei dati storici e sul feedback degli utenti, gli utenti non possono favorire dei reali vantaggi del carsharing se le automobili rimangono inutilizzate in queste zone, dove il tempo di inutilizzo è 4 volte più alto rispetto alla media delle zone altamente frequentate, qui [27] [28] le fonti della notizia.

Figura 4.2: Area operativa del servizio originaria

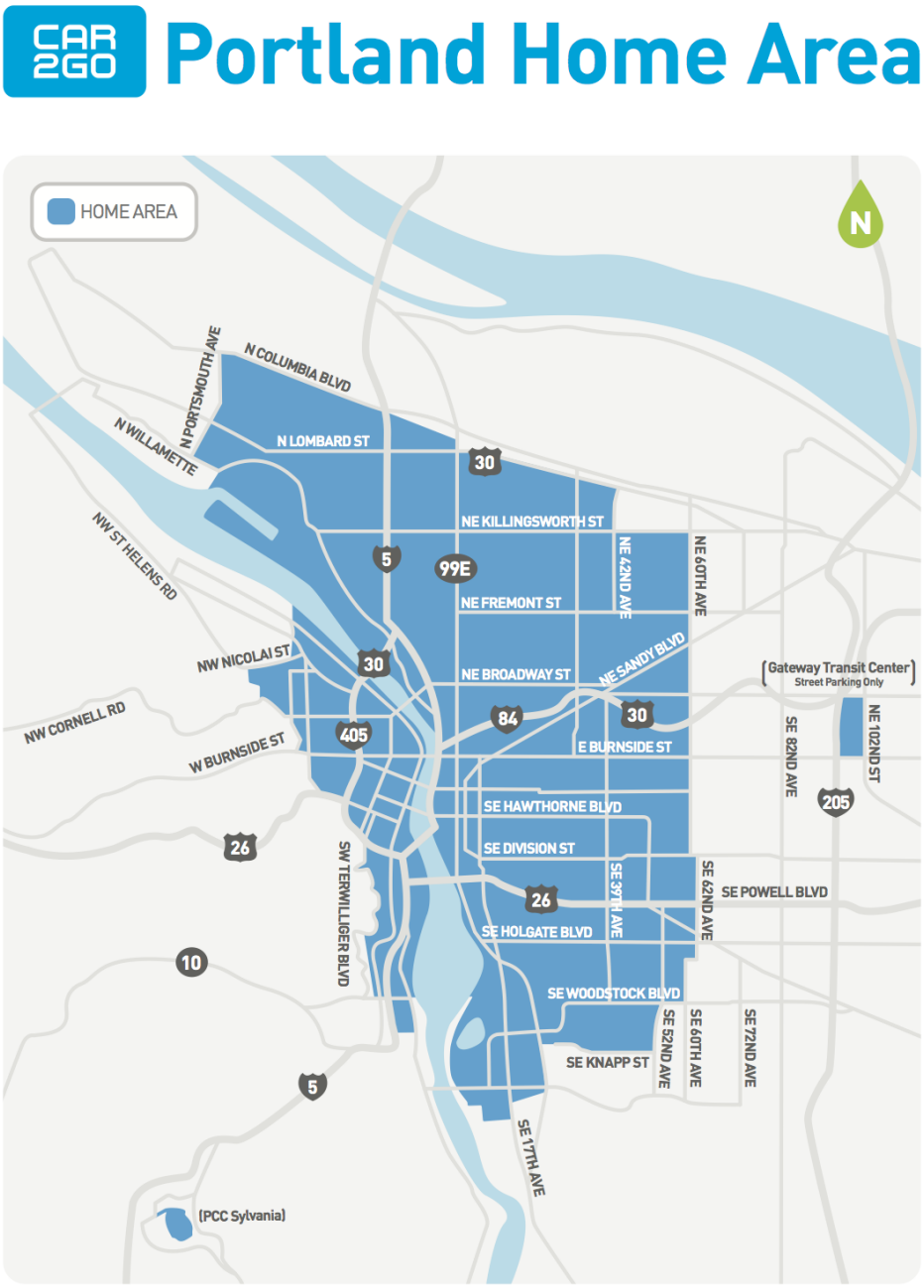


Figura 4.3: Area operativa del servizio modificata a partire dal 24 agosto 2015

4.2.2 Grafici

Analisi globale

La prima analisi svolta ha valutato l'occupazione delle celle per ogni settimana dell'anno, in modo da avere una visione globale dell'utilizzo del servizio. I dati sono stati rappresentati mediante boxplot, così da visualizzare nei dettagli l'andamento del valore osservato. In verde sono evidenziati i valori delle medie.
Come esempio si possono osservare i grafici in figura 4.4.
La cella 52, oltre ad avere un alto numero di veicoli che la occupano, ha un'alta variabilità di occupazione, che raggiunge il suo massimo nella 33° settimana, con dei valori di occupazione che oscillano da 0 a 54. Il caso in cui si ha la variabilità più bassa si verifica nella 4° settimana, dove i valori di occupazione oscillano solamente fino a 10.
Si noti l'andamento dell'occupazione per la cella 37, in settimane come la 19° e la 27° il numero di occupazioni è stato in media di 5/6 veicoli per ogni timestamp, ma con una bassissima variabilità, ciò porta ad ipotizzare che in quelle settimane ci sia stata una stagnazione di veicoli in quella cella.
Tornando a focalizzare l'attenzione sulla cella 52, ci si accorge che in media ha sempre un valore di occupazione intorno al valore 5.
Con questa granularità temporale è difficile trarre delle osservazioni dettagliate, ma si è incominciato a capire quali celle sono più dinamiche e quindi più ostiche dal punto di vista predittivo.

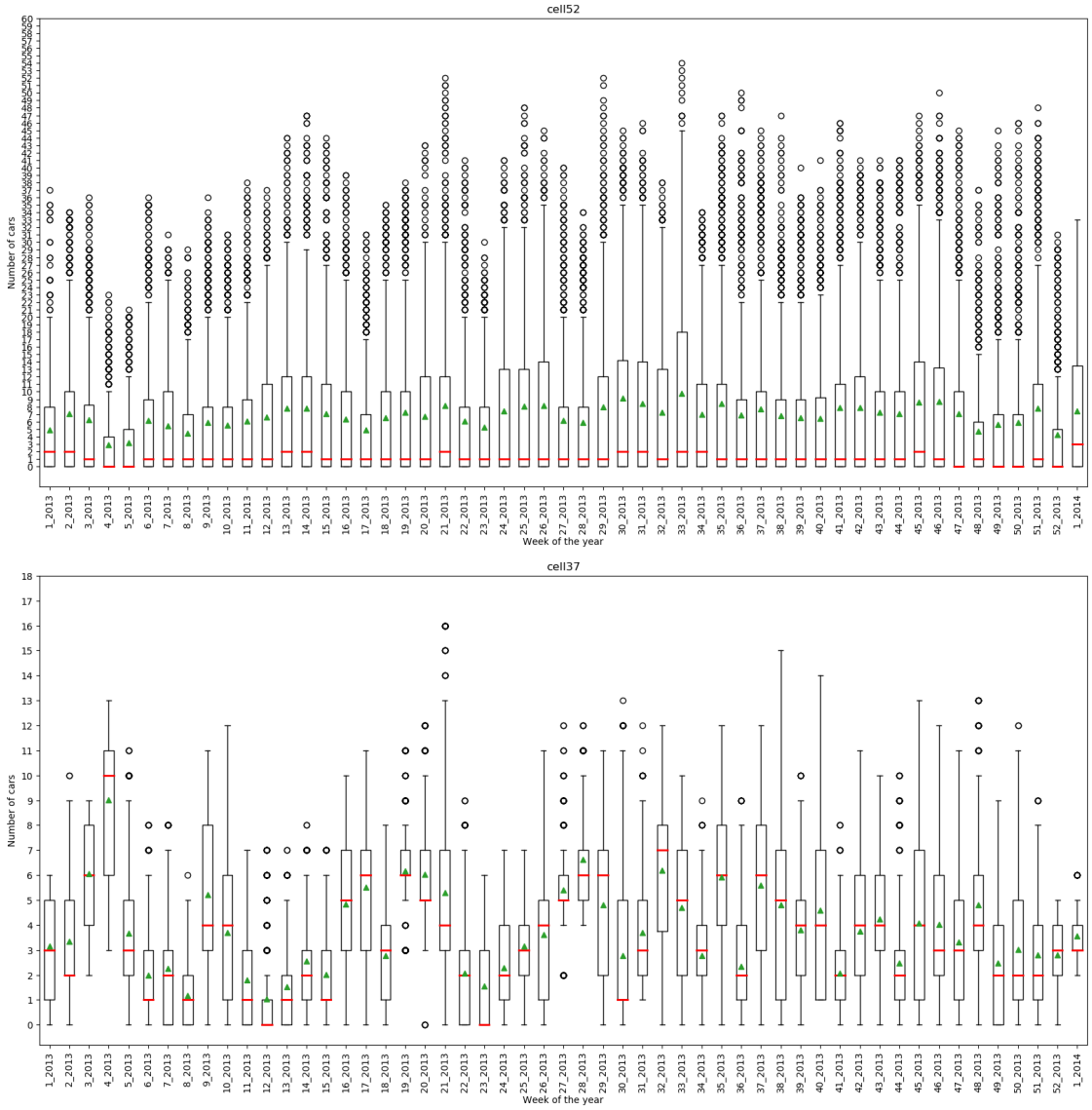


Figura 4.4: Boxplot delle settimane delle celle 52 e 37

Analisi settimanale per timeslot

Con questa analisi si vuole esaminare l'andamento medio dell'occupazione delle celle durante la settimana, aggregando i dati a livello di timeslot (un timeslot ha una durata di 3 ore).
Osservando il grafico per la cella 56 (figura 4.5) si può notare che il numero di veicoli ha un andamento periodico: cresce con l'avvicinarsi delle ore notturne per poi diminuire con l'arrivo delle ore diurne. Questo comportamento è più marcato nei giorni infrasettimanali, mentre per il sabato e la domenica è attenuato, con una maggior disponibilità di veicoli durante le ore diurne.

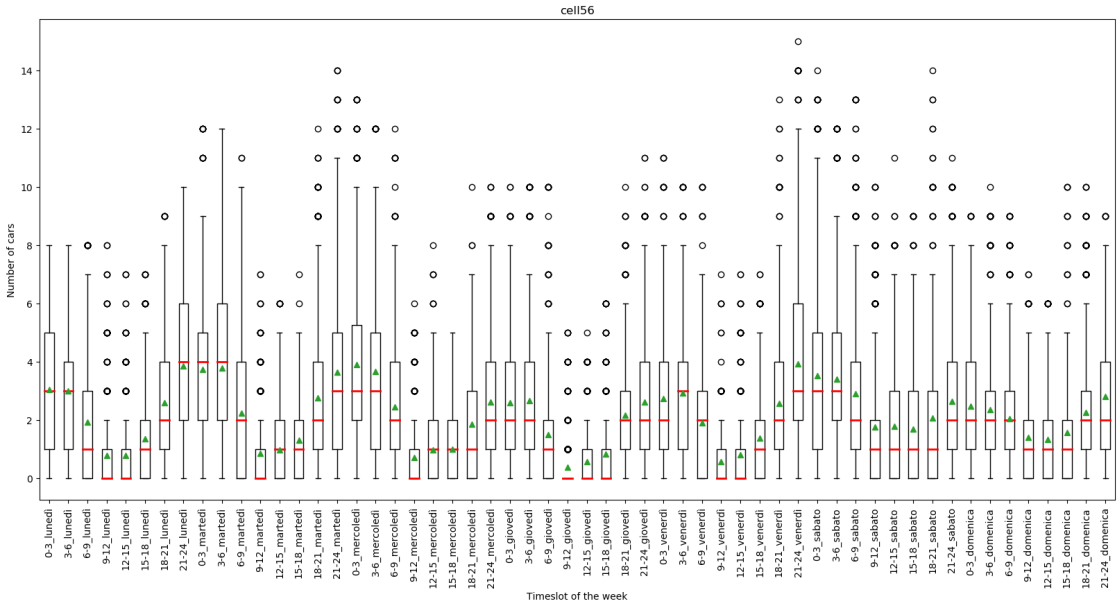


Figura 4.5: Boxplot timeslot della settimana, cella 56

Analizzando la cella 53 (figura 4.6) invece, si rileva un comportamento anch'esso periodico ma opposto a quello della cella 56, l'occupazione della cella raggiunge il suo massimo durante i timeslot '12-15'. Nel weekend, come per il caso precedente, i valori sono smorzati.
Lo stesso comportamento si ritrova anche per le celle numero 49 e 50.
La realizzazione più estrema di questo trend si ottiene con la cella 52 (figura 4.7), con picchi molto alti e variabili di occupazione durante il giorno, dove però il timeslot con l'occupazione maggiore è quello delle ore '9-12' e non quello delle ore '12-15'.

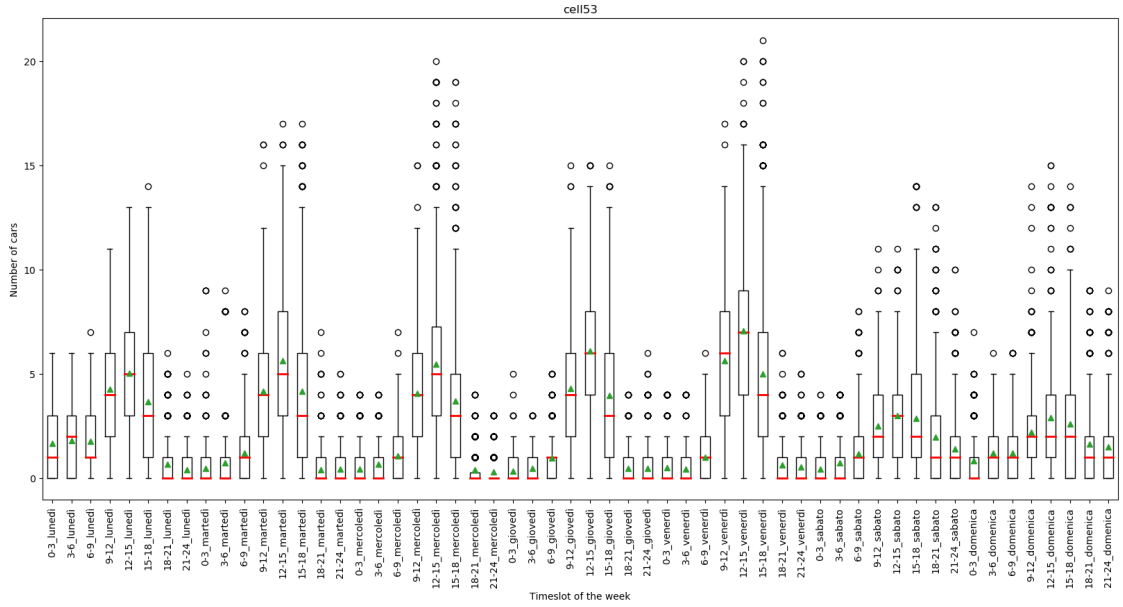


Figura 4.6: Boxplot timeslot della settimana, cella 53

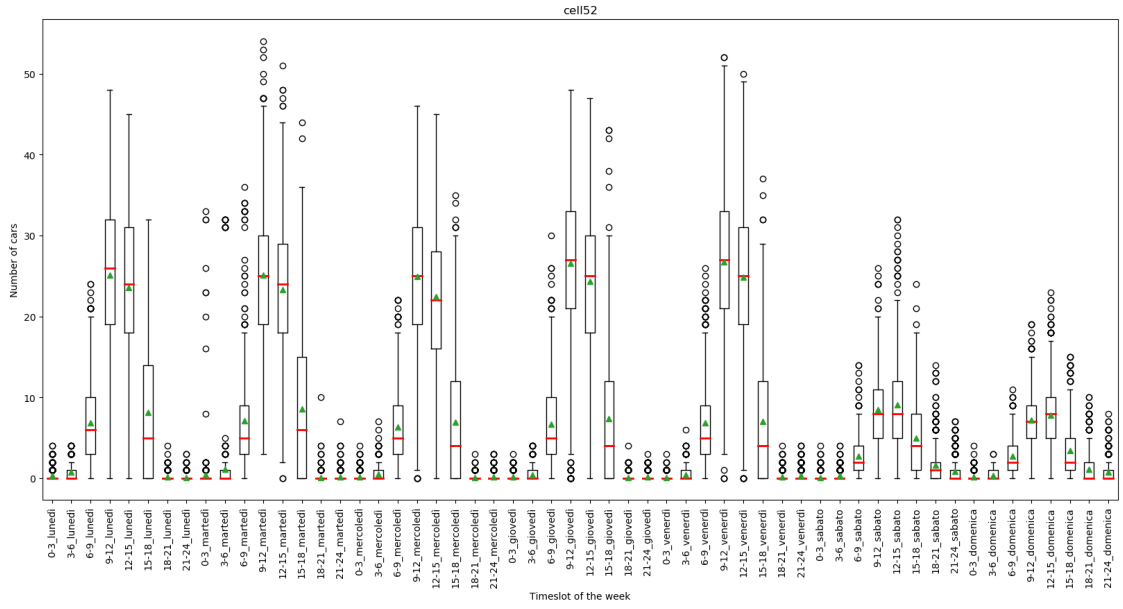


Figura 4.7: Boxplot timeslot della settimana, cella 52

Analisi quotidiana

In seguito sono stati prodotti dei grafici che illustrino la distribuzione media dell'occupazione delle celle per ogni ora distinta del giorno (figura 4.8 e 4.9).

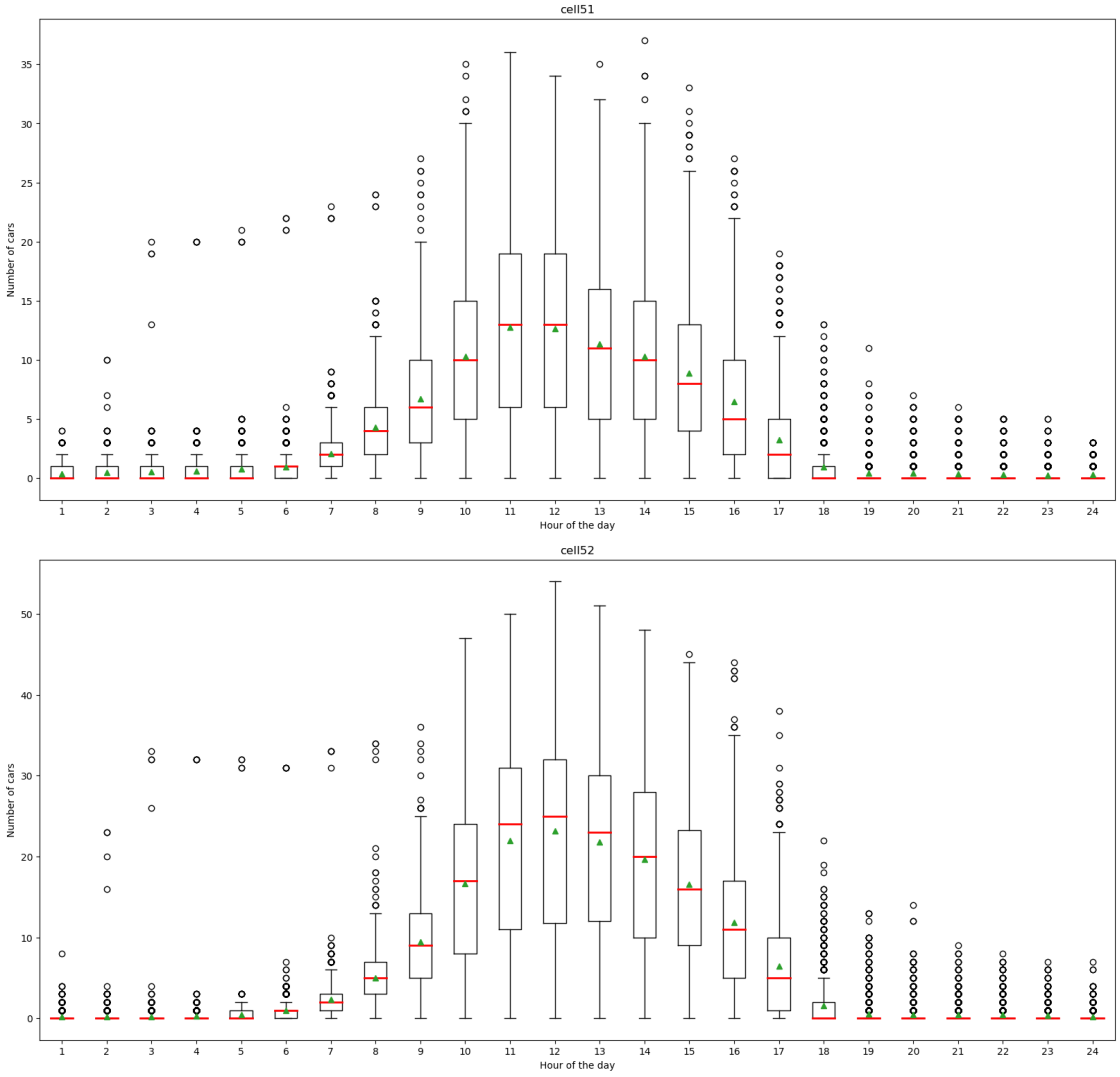


Figura 4.8: Boxplot quotidiani delle celle 51 e 52

Per la cella 51 è immediato notare come dalle ore 7 del mattino il valore medio di occupazione vada a crescere fino al raggiungimento delle ore 11, oltrepassate le ore 12 il valore decresce fino alle ore 18. L'andamento del valor medio si ritrova anche nella variabilità dello stesso, più alta nelle ore centrali della giornata e più bassa durante la notte.
La cella adiacente, la 52, dimostra anch'essa un comportamento simile, ma con valori più alti.

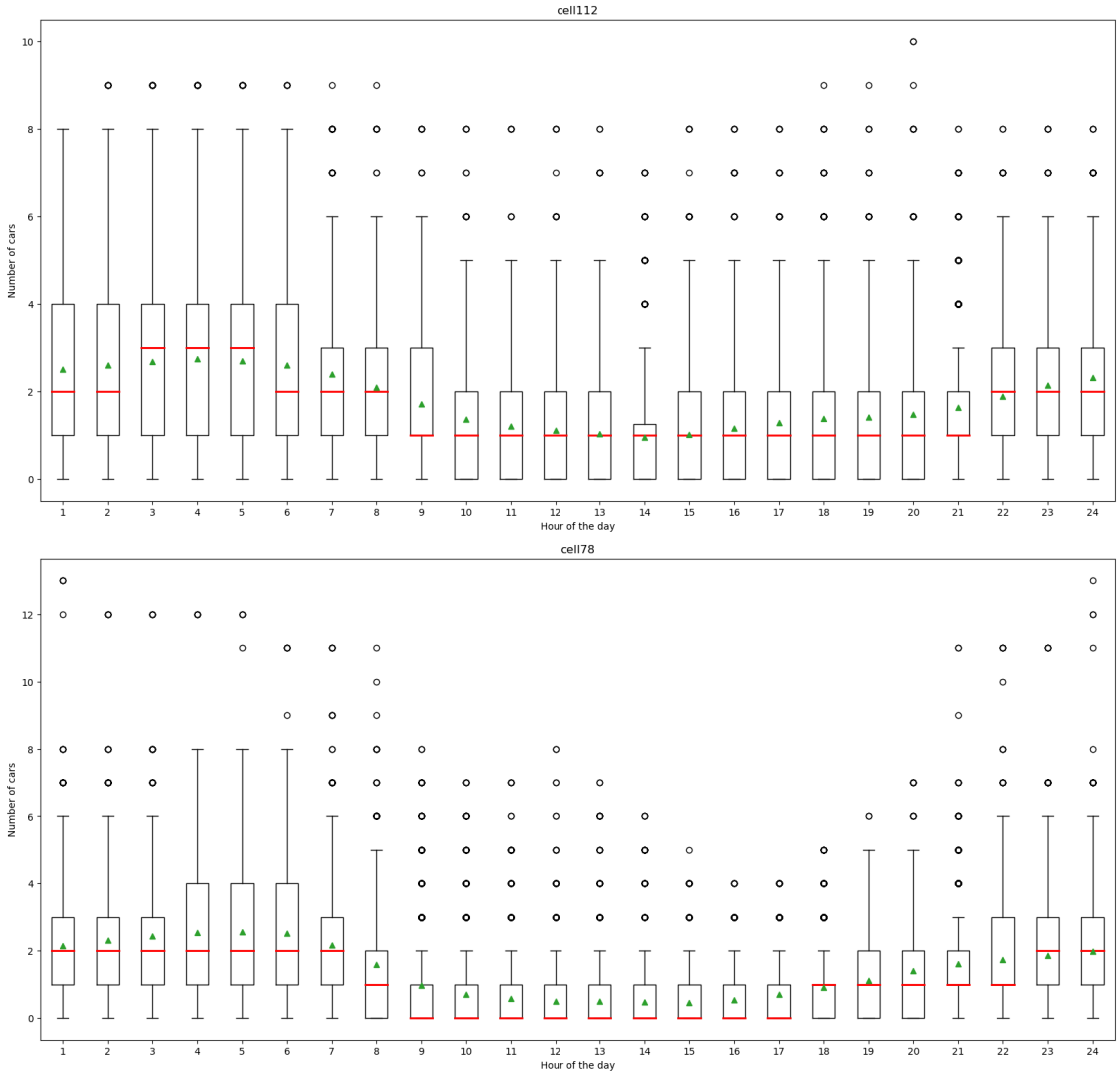


Figura 4.9: Boxplot quotidiani delle celle 112 e 78

La cella 112 invece ha un comportamento opposto ma molto più attenuato, infatti si nota una lenta diminuizione dell'occupazione media tra le ore 6 e le ore 13 ed un ancor più lento incremento nelle ore successive.
Anche la cella 78 subisce un calo dell'occupazione in orario mattutino, ma in un arco di tempo più concentrato rispetto alla cella 112, dalle ore 6 alle ore 10.
Queste osservazioni hanno portato a riconoscere dei pattern tipici d'utilizzo del servizio che identificano le celle come commerciali o residenziali.
Le celle 112 e 78 sono classificabili come residenziali, sono celle in cui durante l'orario mattuttino le autovetture vengo utilizzate dagli utenti per recarsi a svolgere le

loro attività, per esempio lavorare, nelle celle classificate come commerciali; superato il primo pomeriggio, il numero di auto nelle celle aumenta a causa degli utenti che ritornano alla loro abitazione, per la cella 78 per esempio, dalle ore 18 si nota questo comportamento.
Le celle commerciali dimostrano invece un trend inverso, con alte quantità di automobili durante l'orario lavorativo e una quantità nettamente inferiore nelle ore notturne.

Dato che l'analisi per timeslot aveva evidenziato una differenza di utilizzo tra giorni feriali e festivi, si è deciso di approfondire meglio lo studio, in particolare si è deciso di analizzare distintamente l'andamento quotidiano nei giorni festivi e nei giorni feriali, inoltre è stata fatta anche una distinzione tra giorni estivi e giorni invernali, come giorni estivi sono stati considerati quelli appartenenti al periodo temporale 15/05 - 16/06, mentre come invernali quelli appartenenti al periodo 15/01 - 16/02.
Si osservino per esempio i grafici relativi alla cella 52 (figure 4.10 e 4.11), che è quella con il volume più alto di occupazioni ed anche la più variabile, nei giorni: estivi-feriali, estivi-festivi, invernali-feriali, invernali-festivi.
La prima caratteristica che si nota confrontando i grafici relativi al periodo estivo è che durante i giorni feriali il numero di automobili è decisamente maggiore rispetto a quello dei giorni festivi. Per i giorni feriali, le medie di occupazione descrivono un grafico che raggiunge rapidamente il suo massimo, in corrispondenza di mezzogiorno, e poi discende altrettanto rapidamente, mentre invece per i giorni festivi si nota che dalle ore 12 alle ore 15 c'è una costanza della media di automobili presenti. La variabilità è molto alta durante le ore diurne in tutti e due i casi, relativamente al caso stesso. Ad esempio per le ore 12, nei giorni feriali, i valori oscillano tra 13 e 47, mentre nei giorni festivi tra 1 e 15.
Spostando l'attenzione sui grafici del periodo invernale risulta chiaro che per i giorni feriali il trend è uguale a prescindere dalla stagione, con la differenza che durante il periodo invernale il numero totale di occupazioni cala, infatti ad esempio alle ore 12 di un giorno feriale invernale ci sono in media 21 automobili, mentre se fosse estivo ce ne sarebbero in media 31.
Non si può dire lo stesso per quanto riguarda i giorni festivi, nel grafico relativo al periodo invernale infatti non si nota quella costanza del valore che si notava per i giorni estivi, ma è chiara una costante discesa del valor medio dopo le ore 11.

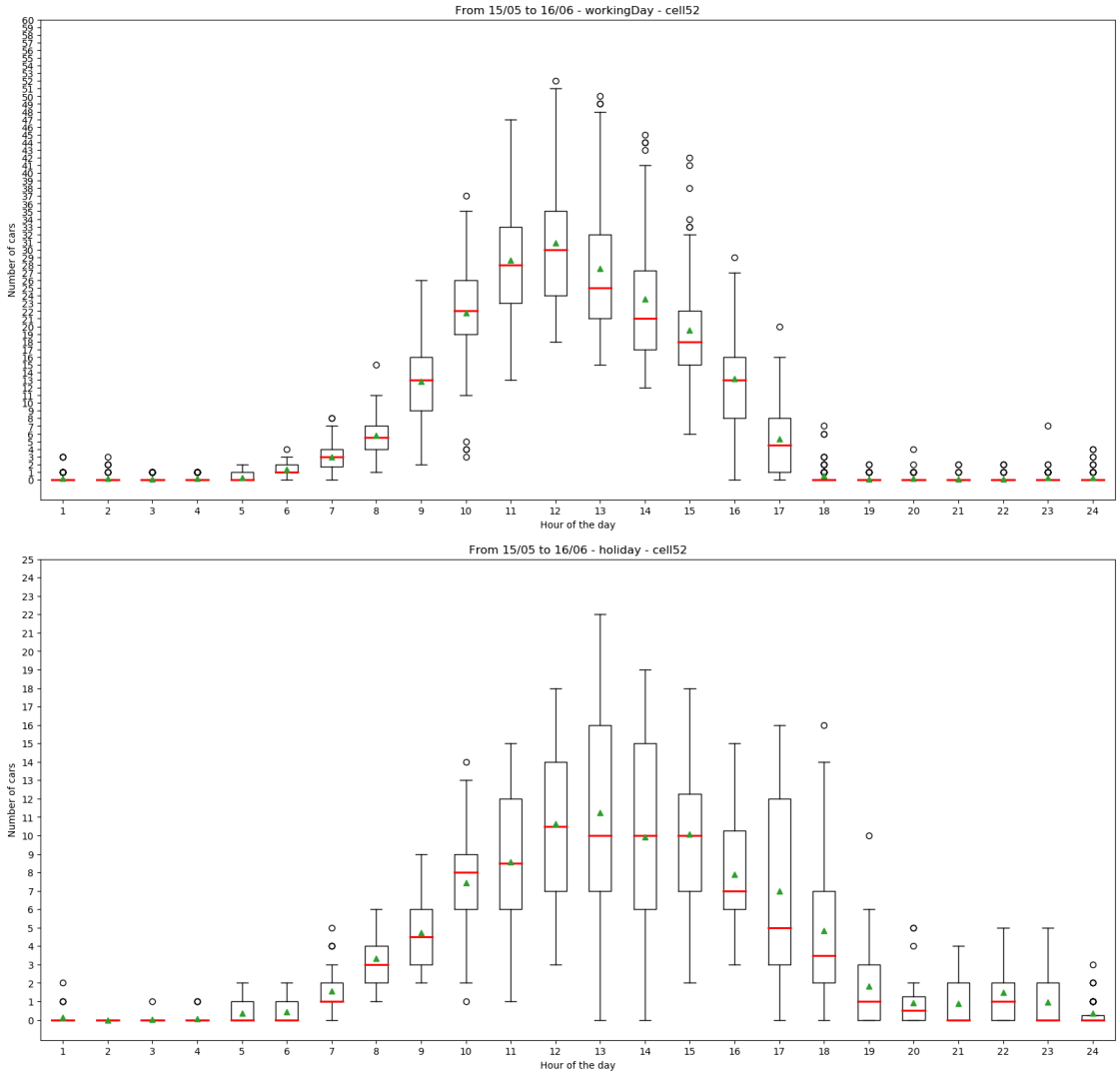


Figura 4.10: Boxplot quotidiani cella 52, periodo estivo, giorni feriali e festivi

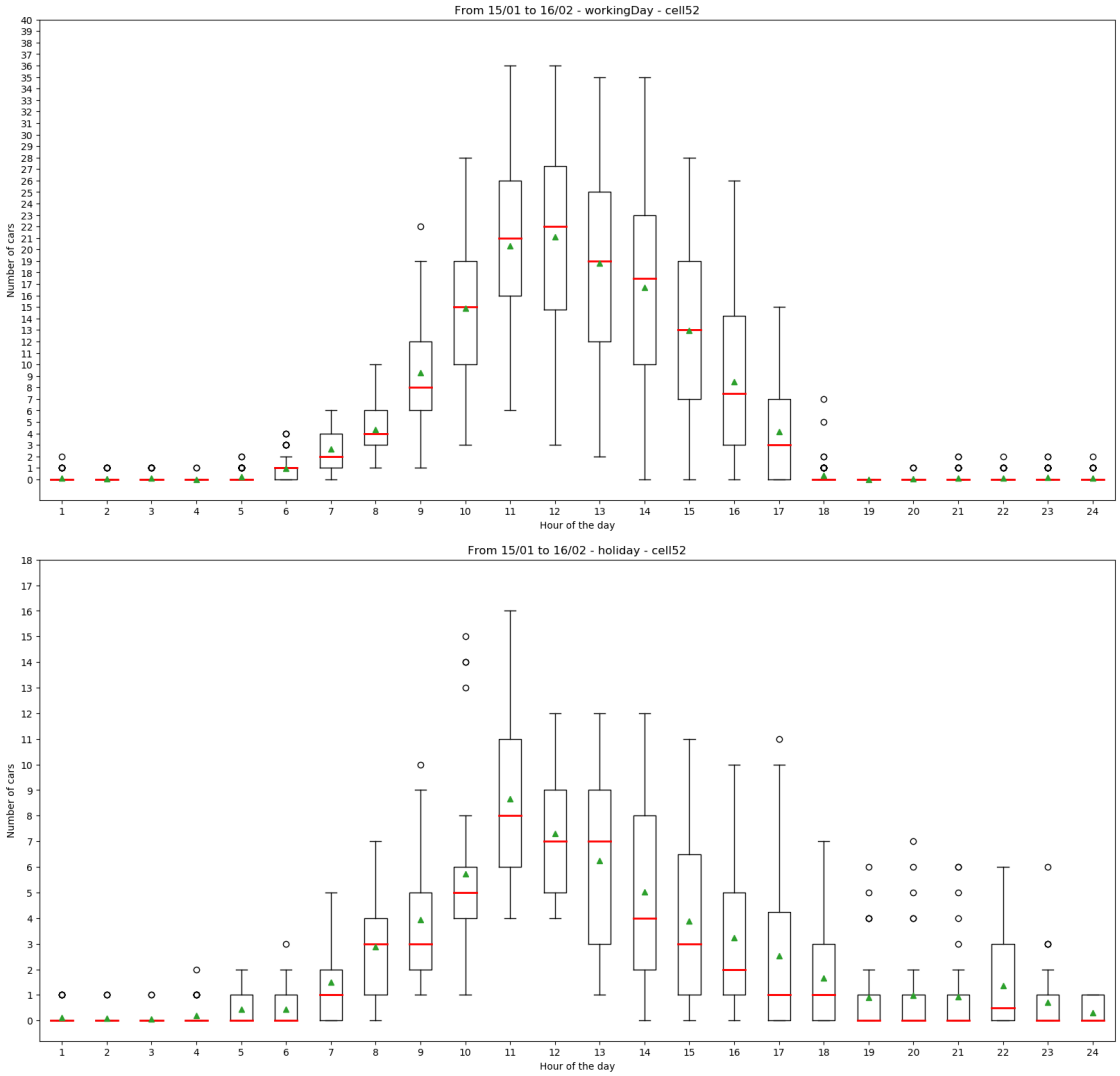


Figura 4.11: Boxplot quotidiani cella 52, periodo invernale, giorni feriali e festivi

Capitolo 5

Valutazione dei modelli predittivi

Con la valutazione dei modelli predittivi si è focalizzata l'attenzione sul confronto delle prestazioni degli algoritmi predittivi al variare dei diversi parametri configurabili.
È stato scelto un sottoinsieme ristretto di celle, tra tutte quelle 'attive', che potesse rappresentare bene il problema, questo vincolo è dipeso dal fatto che i processi predittivi impiegano molto tempo per l'elaborazione. Questo sottoinsieme è composto dalle celle identificate dai numeri 52, 51, 71 e 72; nel corso dell'analisi questo sottoinsieme è stato ridotto ulteriormente.
Le celle selezionate sono celle che presentano sia un elevato numero di presenze di automobili nell'arco dell'anno (come evidenzato nella heatmap 4.1), sia un'elevata variabilità del numero di automobili nei distinti timestamp (come evidenziato dai grafici dell'analisi spaziotemporale 4.2.2), si noti che queste celle corrispondono alla zona centrale della città; sono stati utilizzati questi criteri di scelta nel tentativo di calibrare il modello sul caso peggiore per difficoltà di predizione.

5.1 Performance globali

Inizialmente sono stati generati dei grafici che presentassero i valori del MAE per ogni i-esima iterazione delle 30, per tutti gli algoritmi in esame. In seguito sono stati prodotti dei boxplot che riassumessero l'andamento del MAE sui 30 giorni per ogni algoritmo.
Per tutti i grafici che saranno proposti e commentati da questo punto in avanti se ne specificheranno i parametri solamente quando varieranno, per comodità del lettore quelli principali si potranno dedurre dal titolo di ogni grafico.
Il parametro *horizon* rimarrà costantemente impostato a 8 timestamp per tutte le

analisi, ciò significa che le predizioni saranno sempre relative all'occupazione della cella due ore dopo dall'istante in cui si effettua la predizione.
Nei grafici saranno anche raffigurate le prestazioni di 'baseline', baseline è un metodo di predizione molto semplice, la cui performance è considerata come l'accuratezza minima sotto la quale i modelli che ci scendono non sono da utilizzare. Baseline identifica come valore predetto semplicemente il valore di occupazione della cella al tempo in cui viene espressa la predizione, che nel dataset corrisponde al valore del campo '#carR1(t-8)'. Per quanto riguarda i raggi ed il loro valore, se non specificato esplicitamente, saranno sempre impostati con i valori 500, 1500 e 2000.
Gli algoritmi verranno nominati per mezzo di sigle; SVRlinear, SVRrbf e SVRsigmoid per Support Vector Machine, LR per Linear Regression, GBRT per Gradient Boosting Regressor Trees, RF per Random Forest Regressor.

5.1.1 Esperimenti iniziali

La prima coppia di predizioni ha riguardato la cella 52, con una window pari a 5 e con due date di inizio predizione differenti, 2012-12-31 e 2013-04-30 (figura 5.1). Per la data 2012-12-31 si nota che gli algoritmi SVRrbf e SVRsigmoid sono i meno

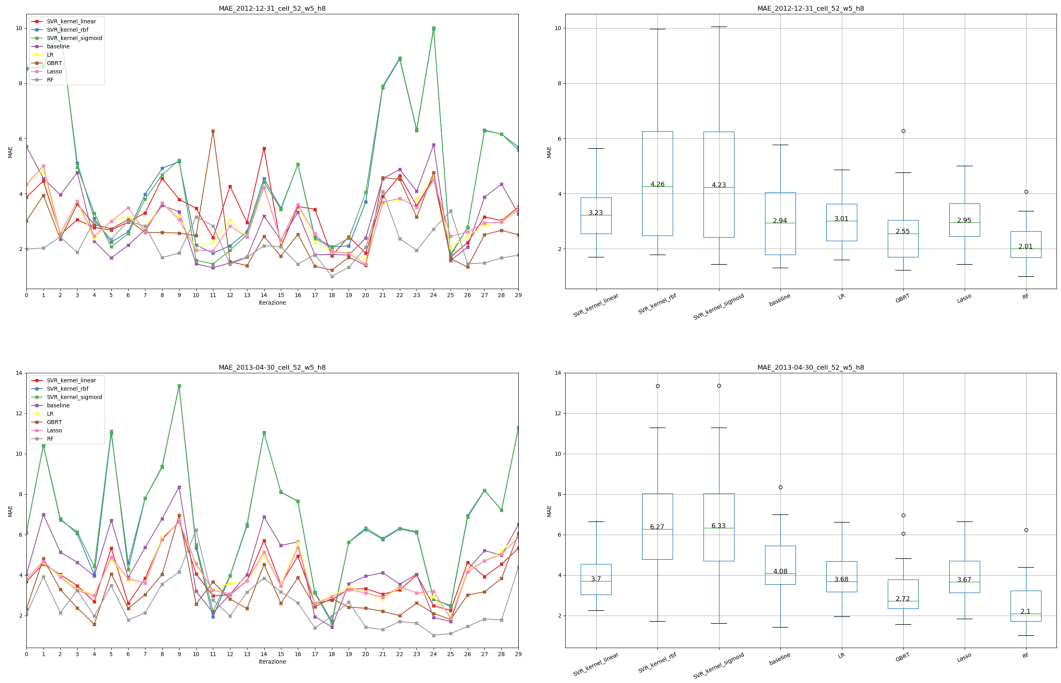


Figura 5.1: Cella 52

accurati, dal boxplot ci si accorge che hanno una media di MAE superiore a 4, gli algoritmi GBRT e RF sono invece quelli che performano meglio.

Eseguendo la predizione partendo dalla data 2013-04-30 si osserva un generale peggioramento delle performance per tutti gli algoritmi, quello che risulta meno colpito dal peggioramento è RF.
La stessa analisi è stata ripetuta per la cella 51 (figura 5.2).

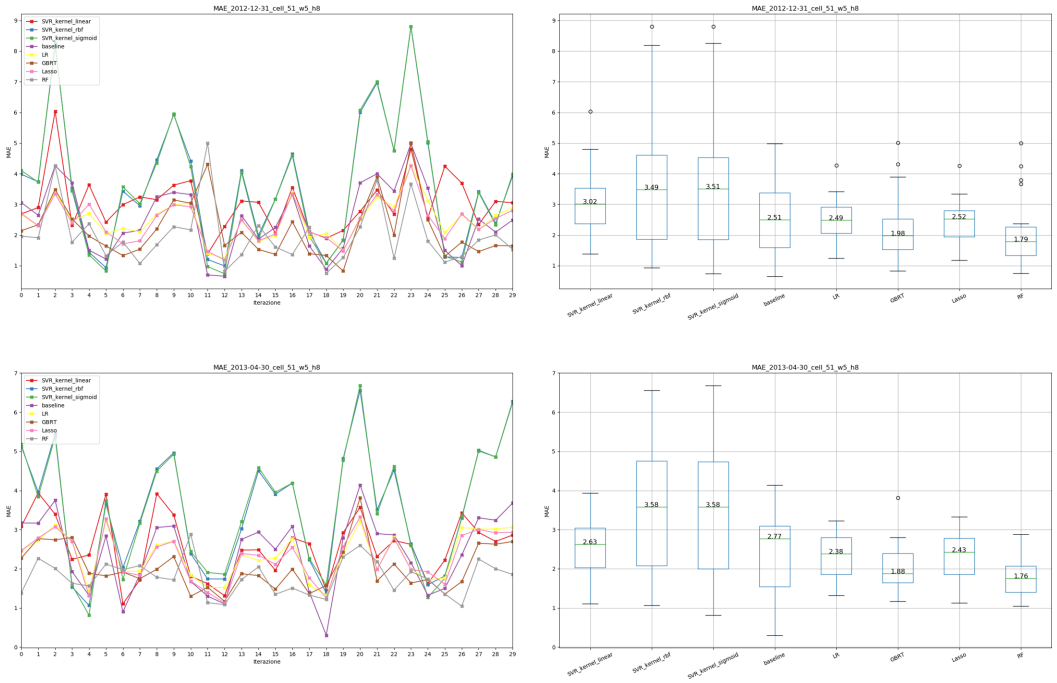


Figura 5.2: Cella 51

Con data 2012-12-31, come per la cella 52, SVRsigmoid e SVRrbf hanno una performance media scadente, mentre GBRT e RF hanno il comportamento migliore.
Con data 2013-04-30 SVRlinear performa meglio di baseline, ma rimangono comunque GBRT e RF i migliori.
Per la cella 72 (figura 5.3) ci si accorge che SVRlinear ha il comportamento peggiore, in tutte e due le date. Le prestazioni per ciascun algoritmo non subiscono variazioni degne di nota al variare del periodo di osservazione.
Anche nelle predizioni con la cella 71 (figura 5.4) SVRlinear risulta il peggiore.
Tra tutti gli algoritmi, sia per la prima che per la seconda data, quello migliore è baseline. Per la data 2012-12-31 è interessante notare che baseline performa mediamente con un MAE inferiore ad 1. Il fatto che baseline performi bene induce a pensare che la cella abbia un andamento piuttosto statico.
Da questi primi risultati appare evidente che la cella 52 sia la più critica per gli algoritmi, osservazione che conferma la previsione fatta dopo la precedente analisi dei grafici riguardanti l'analisi spaziotemporale dell'utilizzo del servizio. La cella 52 è quindi oggetto di ulteriore analisi, osservando come influisce la variazione del

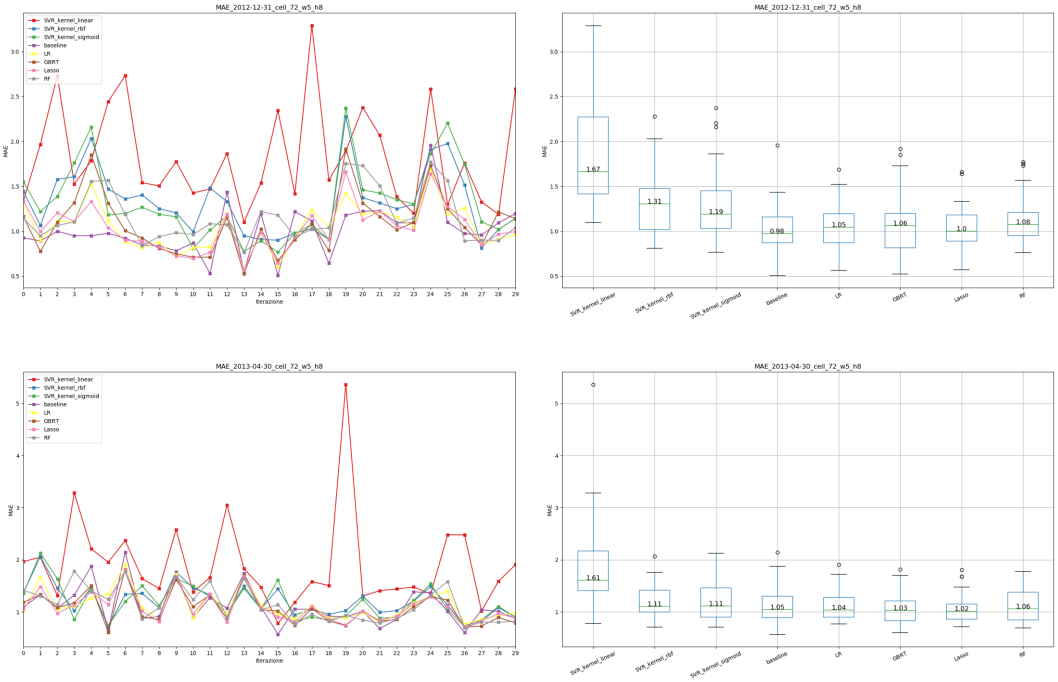


Figura 5.3: Cella 72

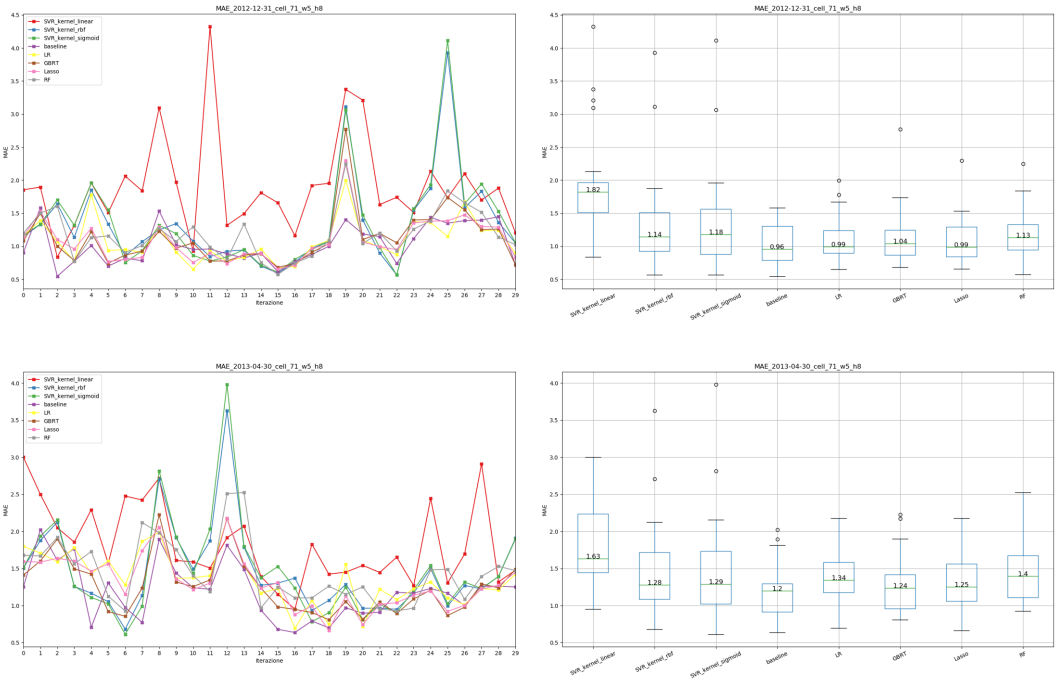


Figura 5.4: Cella 71

parametro window sulla bontà della predizione.
I valori di window considerati sono 3, 5 e 10 (figura 5.5).

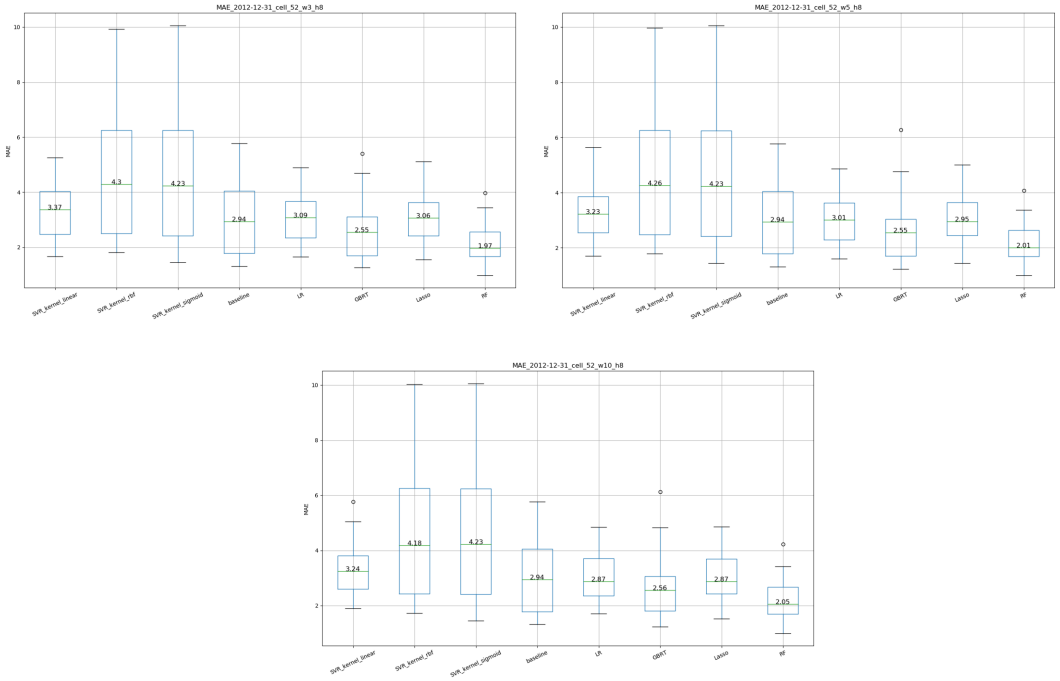


Figura 5.5: Cella 52 con window 3, 5 e 10 e data 2012-12-31

I grafici dettagliati sono stati omessi perchè molto simili.
Dai boxplot si può notare che SVRlinear è l'algoritmo più sensibile alla modifica del parametro, con un abbassamento dell'errore all'aumentare del valore del parametro; baseline, a rigor di logica, non modifica il proprio comportamento; per LR e Lasso l'incremento del valore di window determina un miglioramento della prestazione con la riduzione del MAE; le performance per GBRT e RF peggiorano in modo trascurabile.
Il miglior risultato è quello ottenuto con RF e window impostata a 3. Dal punto di vista del tempo computazionale RF è però anche il più dispendioso, di seguito un confronto tra i tempi di esecuzione di GBRT e RF (figura 5.6).

	3	5	10
GBRT	00:03.343648	00:03.562391	00:04.359239
RF	04:41.893337	05:59.551430	09:22.920168

Figura 5.6: Tempi di esecuzione degli algortimi GBRT e RF con window 3, 5 e 10

L'analisi dell'influenza del parametro window è stata ripetuta per la data 2013-04-30, in questo caso gli algoritmi SVRrbf e SVRsigmoid sono stati omessi dalla visualizzazione perchè poco interessanti (figura 5.7).

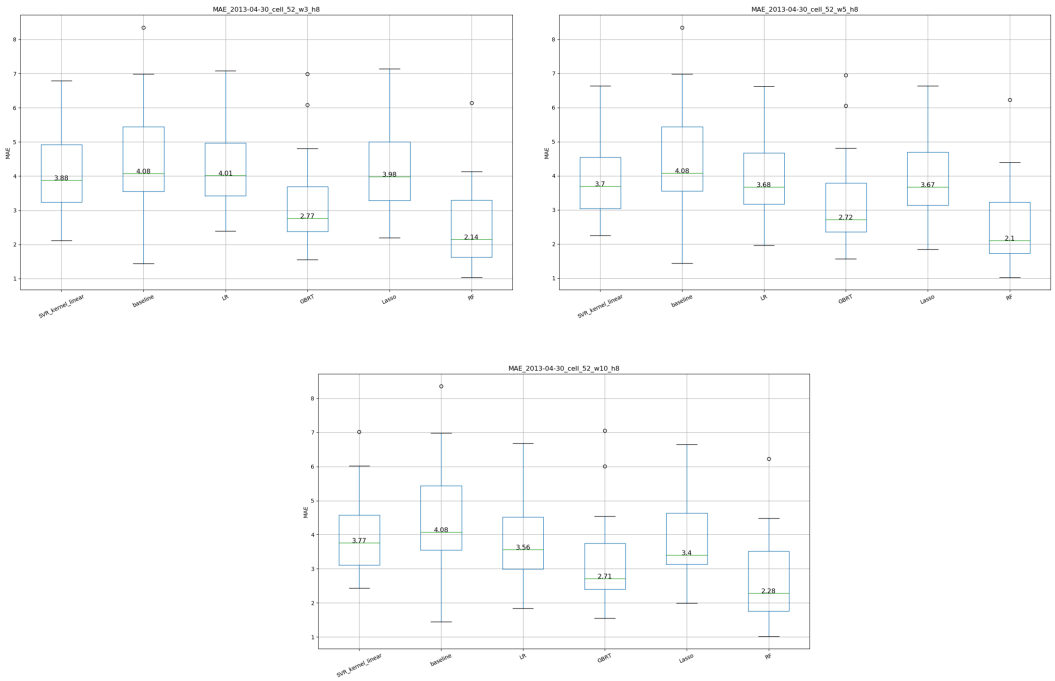


Figura 5.7: Cella 52 con window 3, 5 e 10 e data 2013-04-30

Si assiste ad un generale decadimento delle prestazioni per tutti gli algoritmi, in confronto a quelle ottenute per la data 2012-12-31.
RF, che partendo dalla data 2012-12-31 restituiva un MAE medio pari a circa 2, ora è peggiorato fino ad un massimo di 2,28 con window impostata a 10. Le variazioni di performance per GBRT e RF, in relazione alla variazione di window, sono trascurabili anche con questa data.

5.1.2 Analisi dell'influenza delle feature relative alle condizioni meteorologiche

Con questa analisi si vuole comprendere quale sia il contributo dato dalle feature riguardanti le condizioni meteorologiche alla performance del modello.
I boxplot in figura 5.8 illustrano le prestazioni degli algoritmi per la cella 52 e data 2012-12-31; al variare del parametro window viene confrontata la versione con e senza feature meteorologiche.

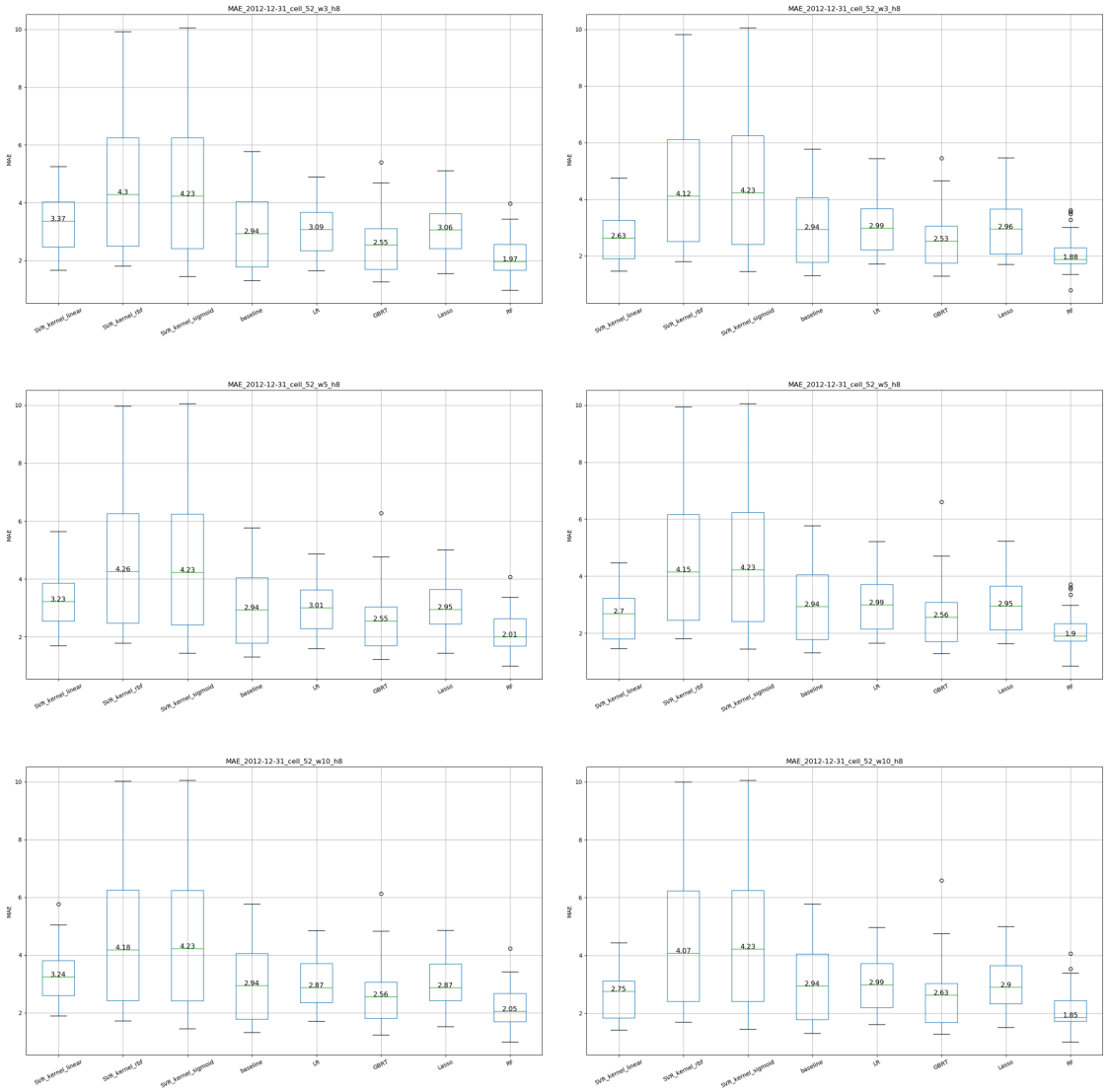


Figura 5.8: Cella 52 con window 3, 5 e 10 e data 2012-12-31, nella colonna di destra la versione senza meteo

L'esclusione delle feature meteorologiche porta ad un incremento della bontà della predizione per l'algoritmo SVRlinear.
Baseline, correttamente, non varia il suo comportamento (l'horizon non cambia).
LR, nel caso di window pari a 3, ha performance migliori nella versione senza meteo, con window pari a 5 circa uguali, mentre con window pari a 10 performa meglio includendo i dati meteorologici.
GBRT non dimostra cambiamenti rilevanti nei casi di window pari a 3 e 5, con window uguale a 10 invece peggiora leggermente senza meteo.
Per Lasso, con window uguale a 3 e 5, si ottiene un leggero miglioramento includendo il meteo, per window pari a 10 non ci sono differenze.
RF, con qualunque valore di window (3 5 e 10), viene penalizzato dall'inclusione delle feature meteorologiche.
Il medesimo set di predizioni è stato ripetuto per la data 2013-04-30, in figura 5.9 sono rappresentati i boxplot che ne riassumono le prestazioni.
SVRlinear, con l'esclusione delle feature riguardanti il meteo, migliora la predizione.
Per LR, nel caso di window pari a 3, il contributo delle feature meteorologiche non è apprezzabile, con window uguale a 5 e 10 invece determinano un leggero miglioramento. Lasso si comporta nello stesso modo di LR.
Le prestazioni di GBRT rimangono pressochè immutate.
RF in tutte le predizioni (window uguale a 3, 5, 10) migliora leggermente senza meteo.
Il comportamento degli algoritmi è analogo a quello osservato per la data 2012-12-31, questa seconda analisi con data 2013-04-30 ha quindi confermato l'influenza che il meteo ha sugli algoritmi.
L'unico algoritmo che ha un giovamento tangibile dall'aggiunta delle feature meteorologiche ed indipendente dall'ampiezza della window è SVRlinear. Per LR e Lasso l'influenza è marginale e, se migliorativa, legata all'aumento dell'ampiezza della window (si ricordi che ad una window più ampia corrisponde una quantità di dati in input al modello più alta). GBRT e RF, identificati finora come i meglio performanti, non ricevono dei benefici dall'introduzione del meteo.
Può essere interessante visualizzare il grafico che viene prodotto dall'algoritmo di feature selection, il cui contributo sarà analizzato nel prossimo capitolo, per comprendere la singola importanza che ha ognuna delle 4 feature meteorologiche sulla predizione. A tal proposito si osservi il grafico (figura 5.10), generato impostando il parametro window a 3, dove è evidenziata la feature 'humidity', che dimostra un'importanza maggiore rispetto a tutte le feature riguardanti la corona circolare del terzo raggio.

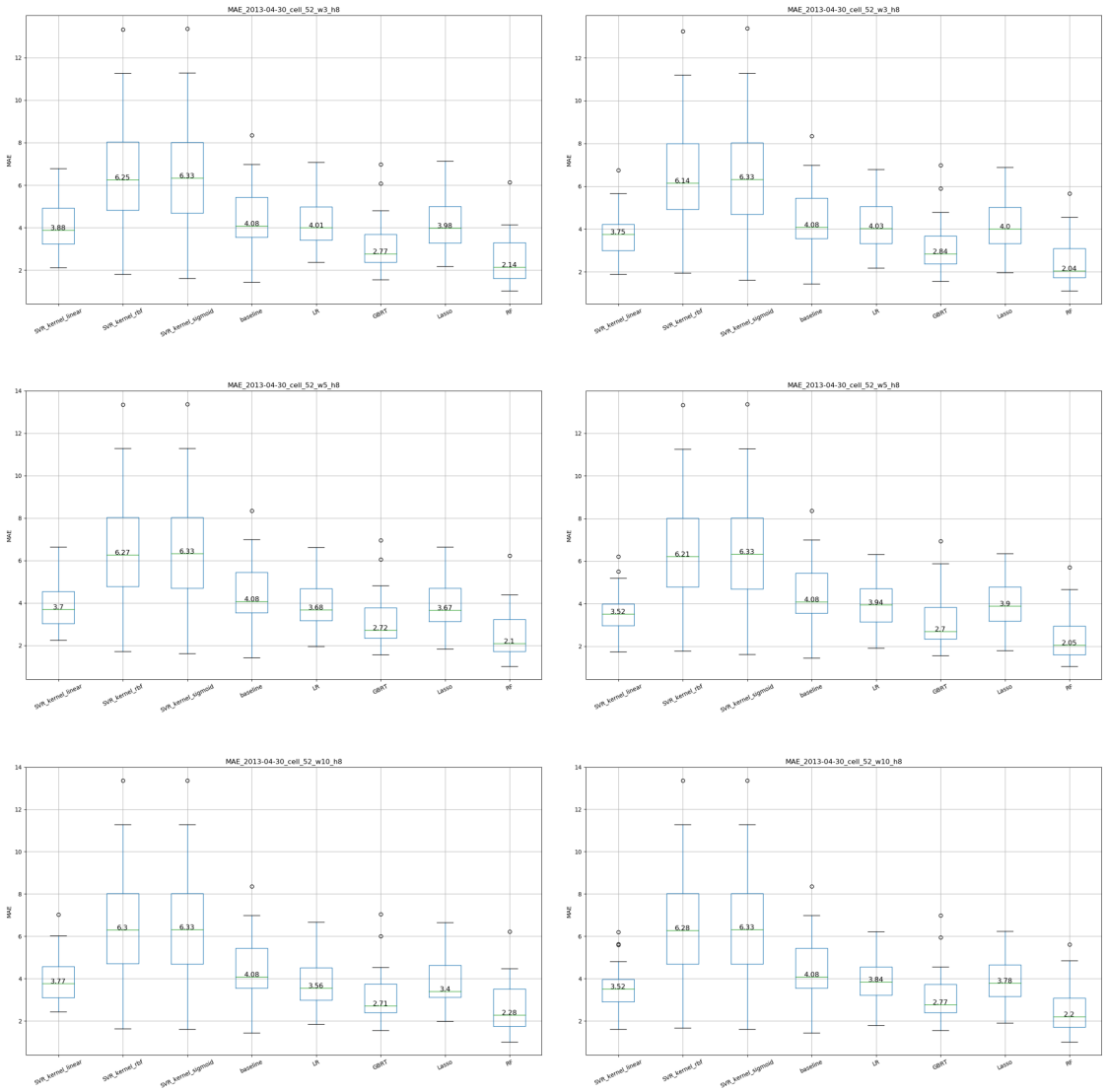


Figura 5.9: Cella 52 con window 3, 5 e 10 e data 2013-04-30, nella colonna di destra la versione senza meteo

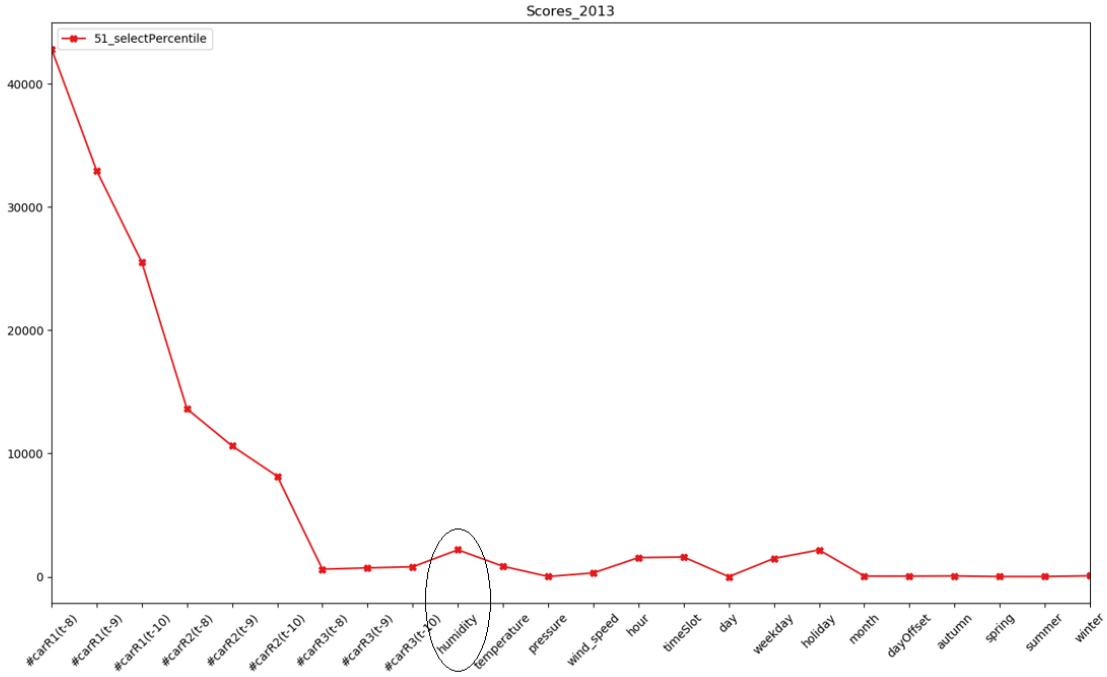


Figura 5.10: Importanza delle feature meteorologiche

5.1.3 Analisi dell'influenza dell'algoritmo di feature selection

Sarà ora valutato l'impatto sull'errore di predizione determinato dall'applicazione di un algoritmo di feature selection al dataset di input.
L'algoritmo di feature selection utilizzato è denominato 'SelectPercentile' e appartiene alla libreria scikit-learn. Per questa valutazione è stato impostato con i parametri 'percentile' pari a 75 e 'score_func' uguale a 'f_regression'. La feature selection viene applicata valutando il dataset corrispondente a tutto l'anno 2013.
Per questa analisi è presa in considerazione la cella 52 in data 2013-04-30.
Nella figura 5.11 i boxplot di sinistra rappresentano le performance delle predizioni effettuate senza feature selection, a destra quelle delle predizioni effettuate con la feature selection attiva, ogni riga si riferisce ad un valore di window diverso (dall'alto verso il basso rispettivamente uguale a 3, 5, 10).
Con window pari a 3 l'algoritmo di feature selection ha eliminato le seguenti features: 'pressure', 'day', 'month', 'dayOffset', 'spring' e 'summer'.
L'algoritmo SVRlinear beneficia della feature selection, gli altri invece non dimostrano miglioramenti, RF grazie alla feature selection perde un outlier presente intorno al valore 6.
Nel caso di window uguale a 5, la feature selection ha eliminato le stesse features del caso precedente più 'winter'.
Come prima, SVRlinear giova di un leggero miglioramento, RF non genera l'outlier pari a circa 6 e gli errori degli altri algoritmi rimangono praticamete immutati.
Le feature eliminate per le predizioni effettuate con window pari a 10 sono uguali a quelle del caso precedente, con l'aggiunta di: '#carR2(t-17)', '#carR3(t-8)', '#carR3(t-9)', 'wind_speed', 'autumn'.
SVRlinear migliora leggermente; LR, Lasso e GBRT non dimostrano variazioni; RF come prima non produce l'outlier.
Si può concludere che per quanto riguarda GRBT la feature selection risulta improducente, mentre su RF elimina un outlier che rappresenta un pessimo valore di MAE, ha quindi un impatto positivo.
Solamente SVRlinear giova di un apprezzabile miglioramento delle prestazioni in seguito all'uso della feature selection.
Nell'immagine 5.12 è visualizzato il grafico che rappresenta l'importanza delle feature, secondo l'algoritmo di feature selection, con window uguale a 10.
È interessante notare che per il terzo raggio le feature eliminate risultano essere '#carR3(t-8)', '#carR3(t-9)', esse sono le più vicine temporalmente al momento della predizione rispetto alle altre ma nonostante ciò vengono eliminate. Questo comportamento potrebbe dipendere dal fatto che i veicoli presenti nella terza corona circolare necessitano di una certa quantità di tempo per spostarsi all'interno della prima circonferenza e influenzare così l'occupazione futura, originando una correlazione tra i dati.

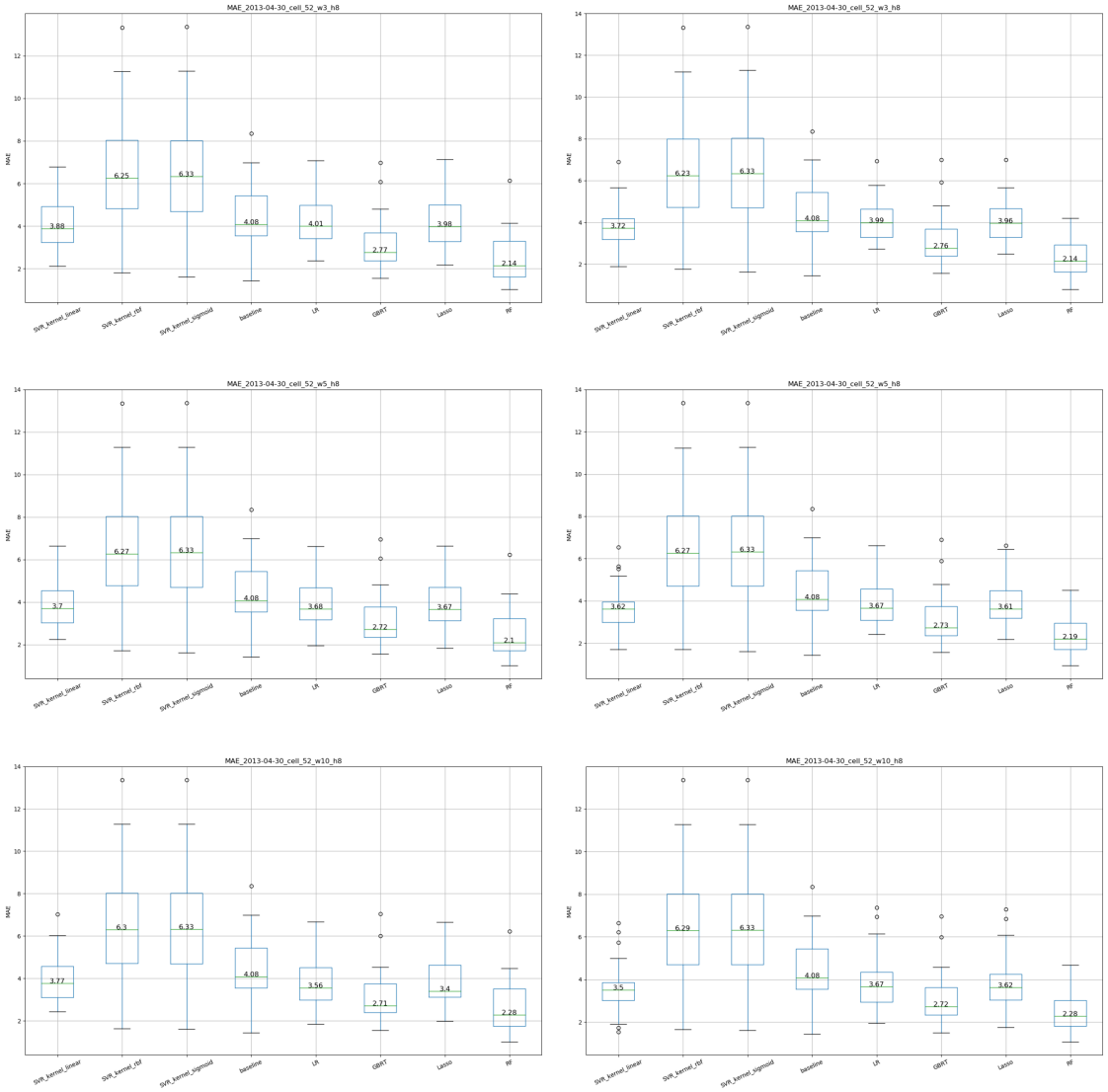


Figura 5.11: Cella 52 con window 3, 5 e 10 e data 2013-04-30, nella colonna di destra la versione con feature selection attiva

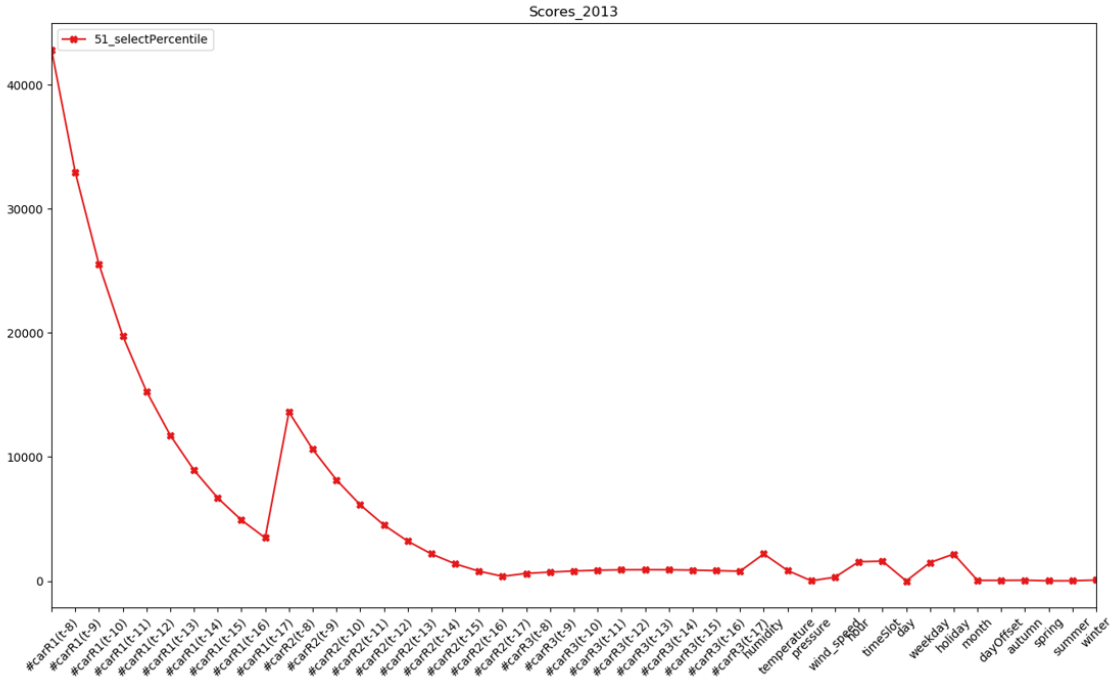


Figura 5.12: Importanza delle feature nel caso di window pari a 10

5.1.4 Differenze di prestazione tra versione assoluta e versione relativa

In questa parte si metterano a confronto le due versioni di modellazione del problema.
Come in precedenza l'attenzione sarà focalizzata sulla cella 52 e data 2013-04-30, agendo sul parametro window e confrontando gli errori tra le due versioni, si considererà anche l'azione della feature selection.
In questa prima figura 5.13 sono confrontate le prestazioni delle due versioni.

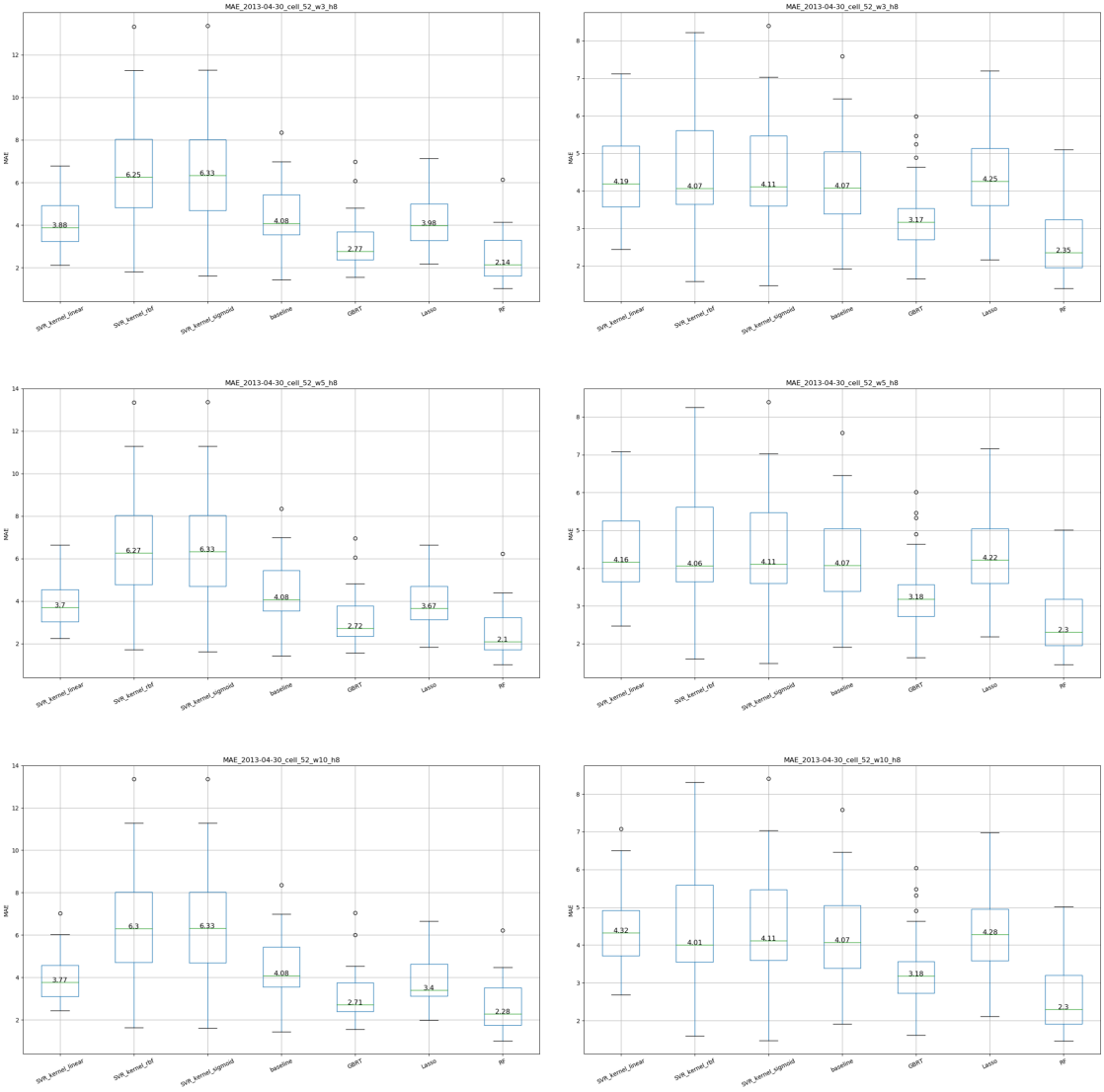


Figura 5.13: Cella 52 con window 3, 5 e 10 e data 2013-04-30, nella colonna di destra la versione 'relativa'

Il boxplot relativo all'algoritmo LR non è visualizzato, perchè nella versione 'relativa' per queste predizioni non converge.
SVRlinear performa meglio nella versione 'assoluta'.
SVRrbf e SVRsigmoid dimostrano un comportamento nettamente migliore nella versione 'relativa', questo fatto è abbastanza rilevante perchè finora questi due algoritmi avevano sempre dimostrato una pessima performance rispetto agli altri algoritmi, in ogni caso RF rimane sempre il migliore, avendo un errore medio circa della metà.
GBRT è più efficiente nella versione 'assoluta', in tutte e due le versioni mantiene la performance costante al variare dell'ampiezza della window, cosa già verificata per la versione 'assoluta'.
Lasso performa meglio nella versione 'assoluta' e all'aumentare del valore di window la predizione diventa più accurata, cosa che invece non avviene per la versione 'relativa'.
RF si comporta come GBRT, ma complessivamente meglio di quest'ultimo.
L'analisi è ora ripetuta con gli stessi parametri ma attivando la feature selection per tutte e due le versioni (figura 5.14).
Con la feature selection l'algoritmo LR non ha problemi nella versione 'relativa', quindi sarà di nuovo visualizzato nei grafici.
Le performance degli algoritmi SVR, GBRT e Lasso hanno lo stesso andamento sia che si attivi la feature selection o meno.
RF performa meglio di GBRT; all'aumentare di window, sia per la versione 'assoluta' che 'relativa', le prestazioni calano.
Fino ad ora la predizione migliore vista in assoluto si ottiene con: algoritmo RF, versione 'assoluta', window=3, feature selection attiva.
Con questa combinazione si ottiene un MAE medio di 2.14, senza outliers.

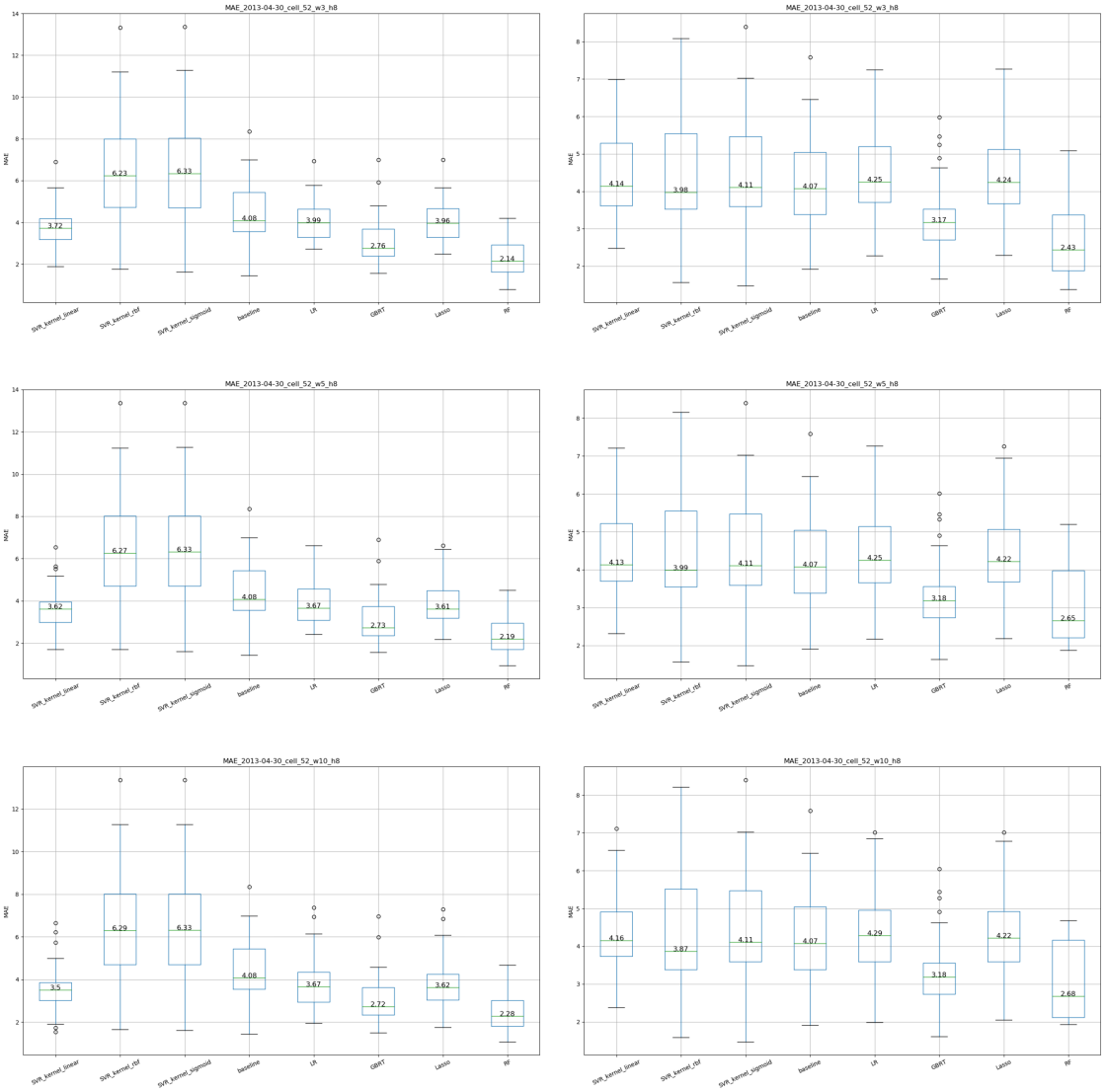


Figura 5.14: Cella 52 con window 3, 5 e 10, data 2013-04-30 e feature selection attiva, nella colonna di destra la versione ‘relativa‘.

5.1.5 Analisi dell'influenza delle feature relative alle corone circolari

Sarà ora analizzato il contributo delle feature riguardanti lo storico dell'occupazione delle corone circolari alla bontà della predizione, che d'ora in poi saranno nominate 2° e 3° raggio.
Nella figura 5.15 sono messe a confronto le performance delle predizioni generate utilizzando lo storico del 2° e 3° raggio (come é stato fatto finora) e quelle generate senza queste feature (nella colonna a destra), i parametri comuni sono: cella 52, data 2013-04-30 e window 3, 5 e 10. Nel caso di window uguale a 3, LR non ha converso e non è quindi visualizzato. Con window pari a 3 tutti gli algoritmi performano praticamente nello stesso modo, Lasso e RF dimostrano un miglioramento quasi impercettibile.
Nel caso di window uguale a 5, gli algoritmi si comportano nello stesso modo appena descritto, con anche SVRlinear che migliora impercettibilmente.
Nel caso di window uguale a 10, SVRlinear performa meglio senza R2 ed R3, LR peggiora leggermente, Lasso e GBRT mantengono circa le stesse prestazioni, RF come SVRlinear migliora.
In media la miglior predizione sembra che si ottenga con l'algoritmo RF, senza R2 ed R3. Il valore di window è quasi ininfluente.

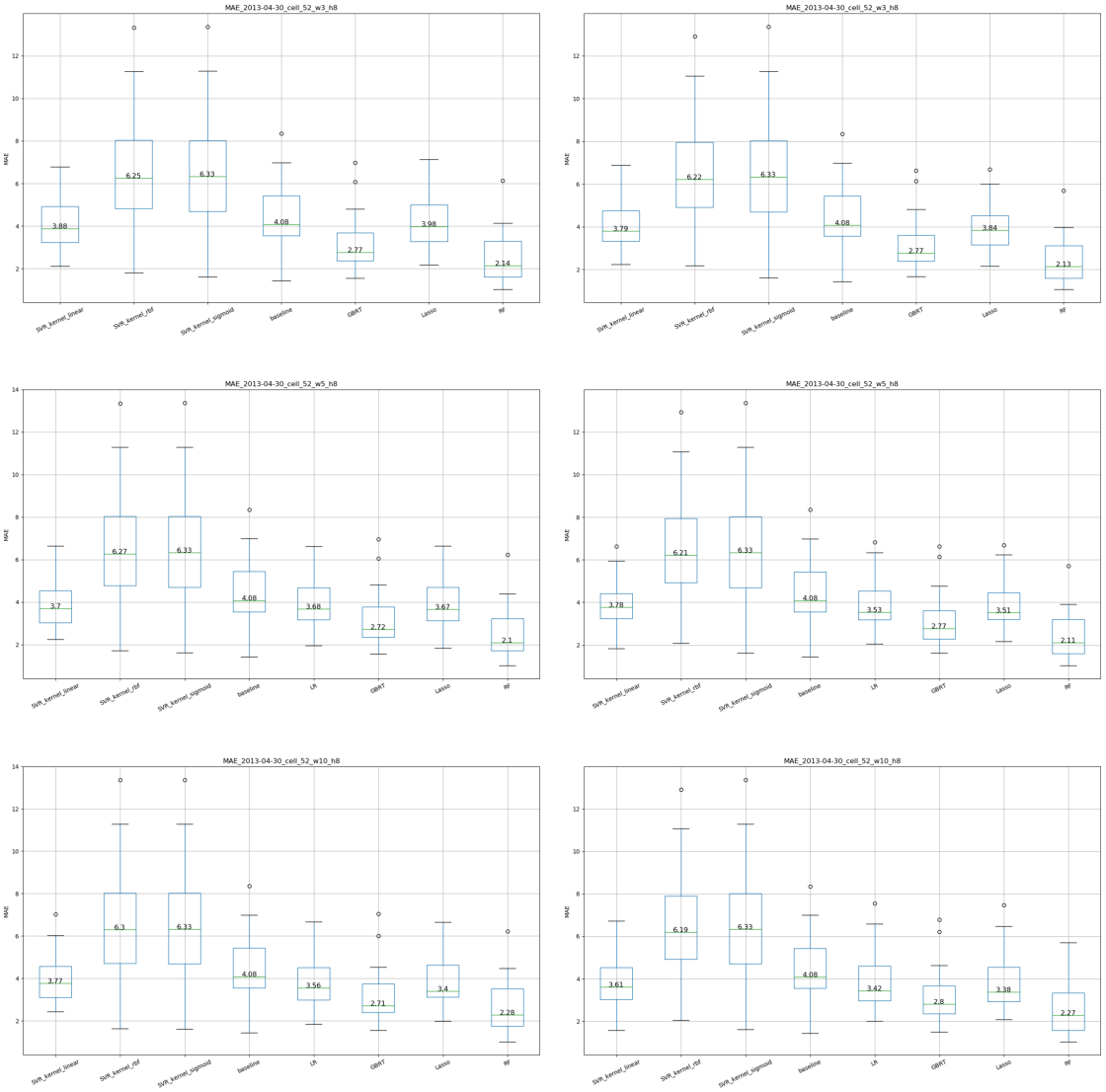


Figura 5.15: Cella 52 con window 3, 5 e 10, data 2013-04-30, nella colonna di destra la versione senza 2° e 3° raggio

5.2 Performance per timeslot

In questa parte di lavoro si osserverà l'andamento del MAE per differenti fasce orarie della durata di 3 ore (0-3, 3-6, ...).
Nei boxplot successivi non saranno più confrontati i MAE per ogni algoritmo, ma saranno confrontati i MAE che un algoritmo genera relativamente ad ogni timeslot.
Come prima analisi si osservino i boxplot degli errori generati dall'algoritmo RF per la cella 52 in data 2013-04-30 con la window impostata a 3, 5 e 10 (figura 5.16).

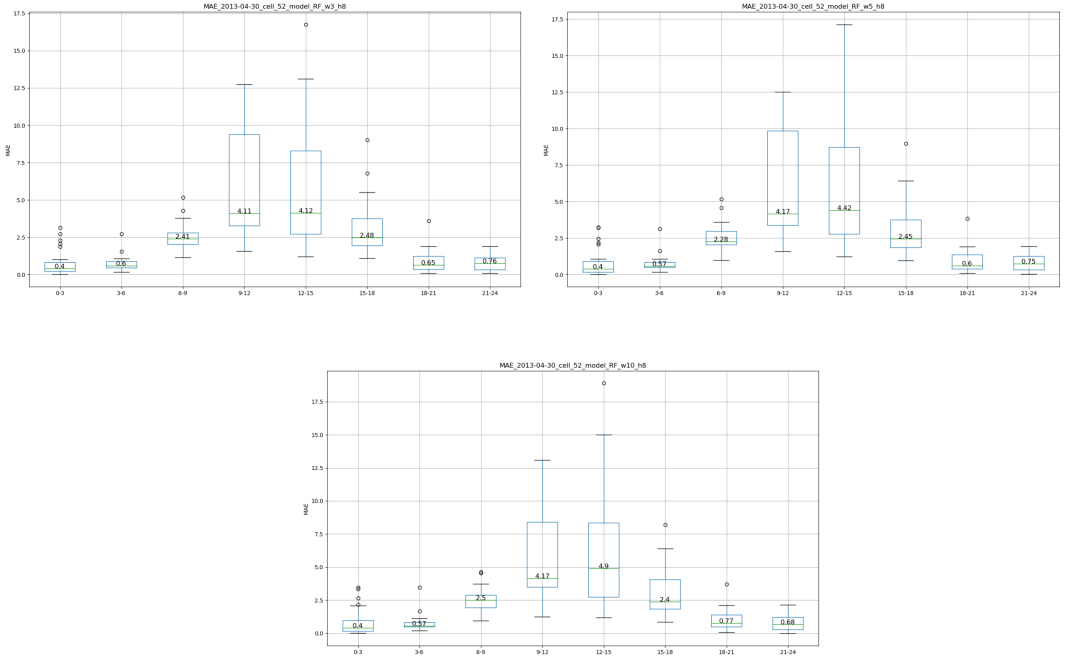


Figura 5.16: Algoritmo RF, cella 52, data 2013-04-30 e window 3, 5 e 10

Dai boxplot si può osservare che esistono fasce orarie più critiche di altre, in particolare i timeslot 9-12, 12-15 ed in misura minore 15-18.
Per il timeslot 9-12 è interessante notare un miglior comportamento di RF con window pari a 5, mentre con window pari a 10 si ha la peggior performance del timeslot 12-15, con un errore massimo incrementato ad oltre 17,5.
In figura 5.17 si può osservare il comportamento di GBRT nelle medesime condizioni.
Per GBRT la variazione del parametro window non causa differenze degne di nota nei risultati.
Facendo un confronto tra i due algoritmi, in particolare riferendosi ai boxplot con window uguale a 10, per i timeslot 0-3, 3-6 e 6-9 gli algoritmi performano in modo

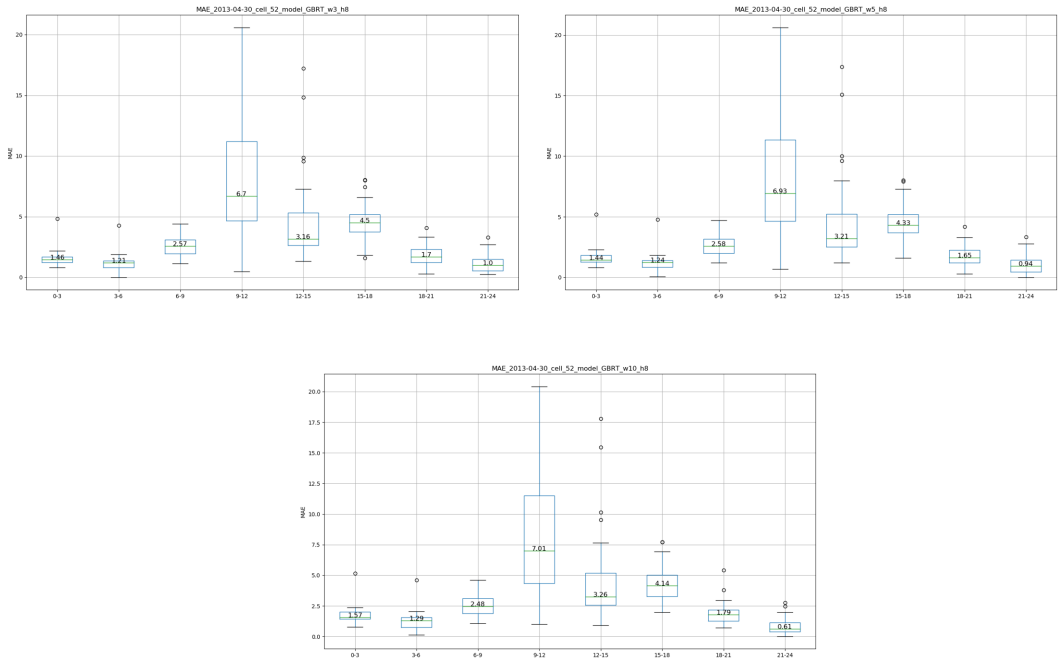


Figura 5.17: Algoritmo GBRT, cella 52, data 2013-04-30 e window 3, 5 e 10

abbastanza simile, per il timeslot 9-12 osserviamo che RF è l'algoritmo migliore, per 12-15 gli outliers nel boxplot dell'algoritmo GBRT rendono difficile il confronto, mentre per 15-18, 18-21 e 21-24 le prestazioni migliori si ottengono con RF.
Vengono ora replicate le osservazioni con le stesse condizioni ma sulla cella 51.
In figura 5.18 le prestazioni per l'algoritmo RF.
Anche per la cella 51 le fasce che risultano maggiormente critiche sono 9-12 e 12-15.
Rispetto alla cella 52, gli errori sono più contenuti. La variazione del valore di window non evidenzia cambiamenti rilevanti nell'entità degli errori.
In figura 5.19 le prestazioni per l'algortimo GBRT.
Agendo su window, la variazione più notevole per GBRT si osserva sulle mediane degli errori del timeslot 9-12 che, con window uguale a 10, migliorano rispetto alla versione con window uguale a 3 e 5.
Confrontando i due algoritmi, in generale RF ottiene delle performance migliori rispetto a GBRT, solamente per il timeslot 12-15 si ha un migliore comportamento di GBRT.
Si analizzano ora i restanti algoritmi sulla data 2013-04-30.
L'andamento degli errori per l'algoritmo Lasso, applicato sulla cella 52 in data 2013-04-30 variando il valore di window, è visualizzabile in figura 5.20.
Possiamo notare che per questo algoritmo migliora leggermente la predizione con l'aumentare del valore di window. Lasso in confronto a RF performa in maniera

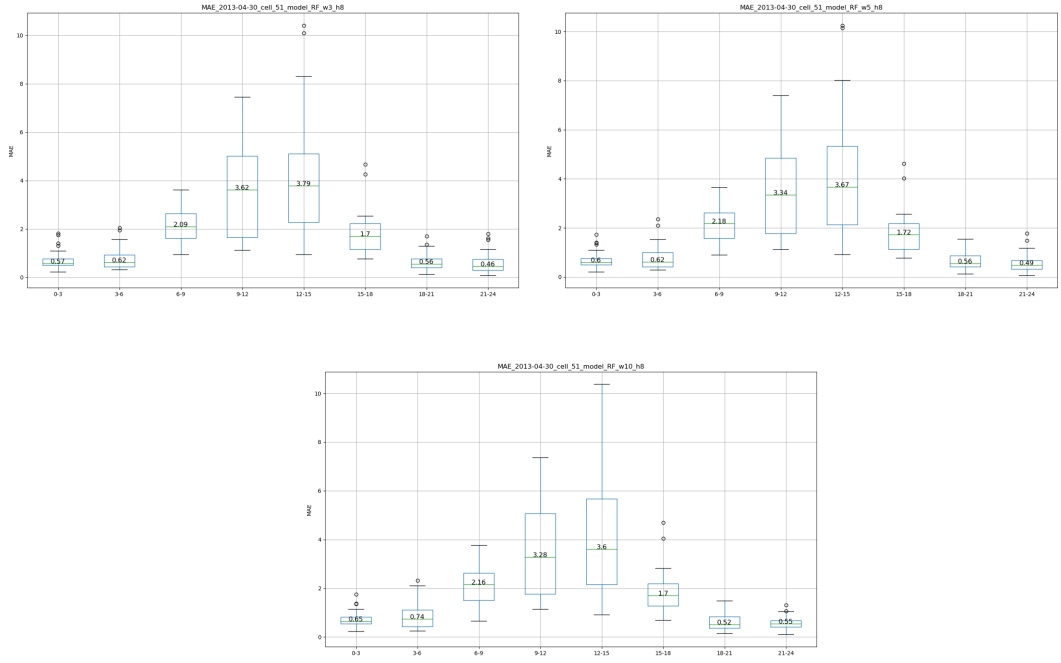


Figura 5.18: Algoritmo RF, cella 51, data 2013-04-30 e window 3, 5 e 10

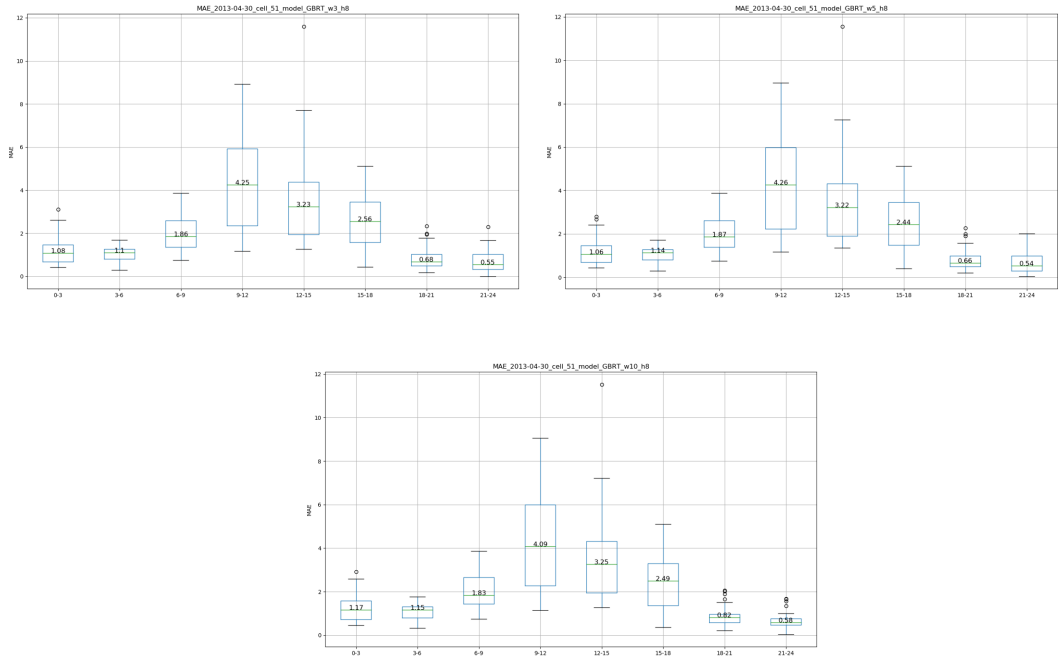


Figura 5.19: Algoritmo GBRT, cella 51, data 2013-04-30 e window 3, 5 e 10

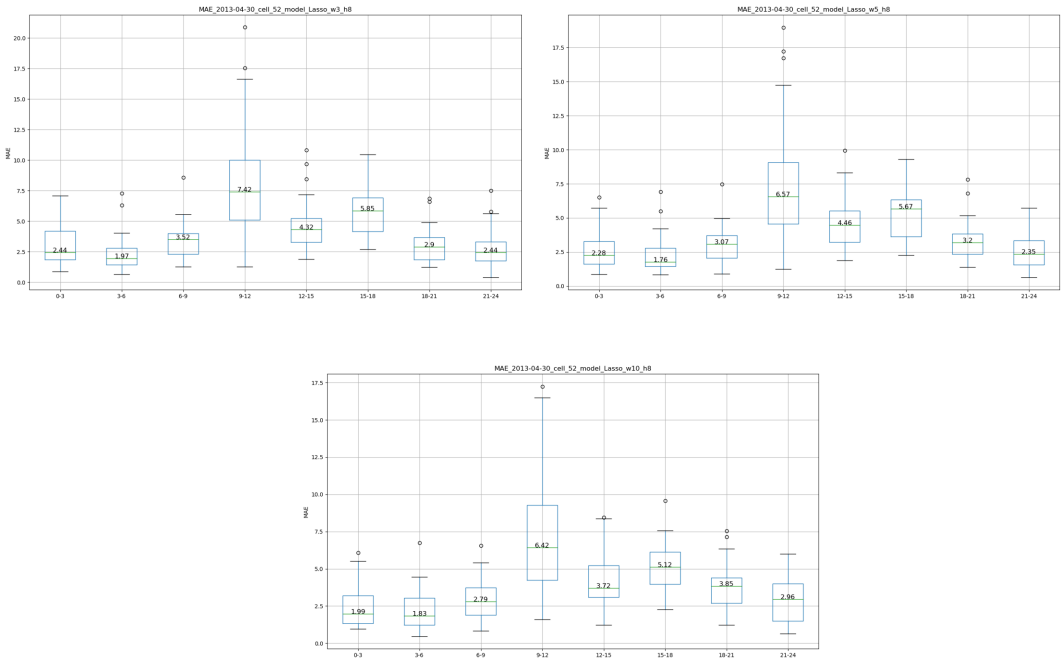


Figura 5.20: Algoritmo Lasso, cella 52, data 2013-04-30 e window 3, 5 e 10

nettamente peggiore. Notiamo una certa criticità per le predizioni del timeslot 9-12, già riscontrata con l'algoritmo GBRT.
Le perfomance per SVRlinear sono illustrate in figura 5.21.
Per SVRlinear l'incremento del valore di window non determina variazioni rilevanti nella predizione. Produce una pessima performance per il timeslot 9-12 e si comporta nettamente peggio rispetto ad RF.
L'analisi prosegue osservando i boxplot, sempre dell'algoritmo SVR, ma con kernel differenti, SVRrbf e SVRsigmoid (figure 5.22 e 5.23). Per SVRsigmoid è visualizzato un solo grafico perchè il parametro window non ha influenza sull'errore.
Come si puo' notare dai grafici, SVR con i suddetti kernel produce delle predizioni molto scadenti. Il parametro window non varia le predizioni prodotte da SVRsigmoid, mentre per SVRrbf l'aumento determina un impercettibile calo delle prestazioni sui timeslot critici.

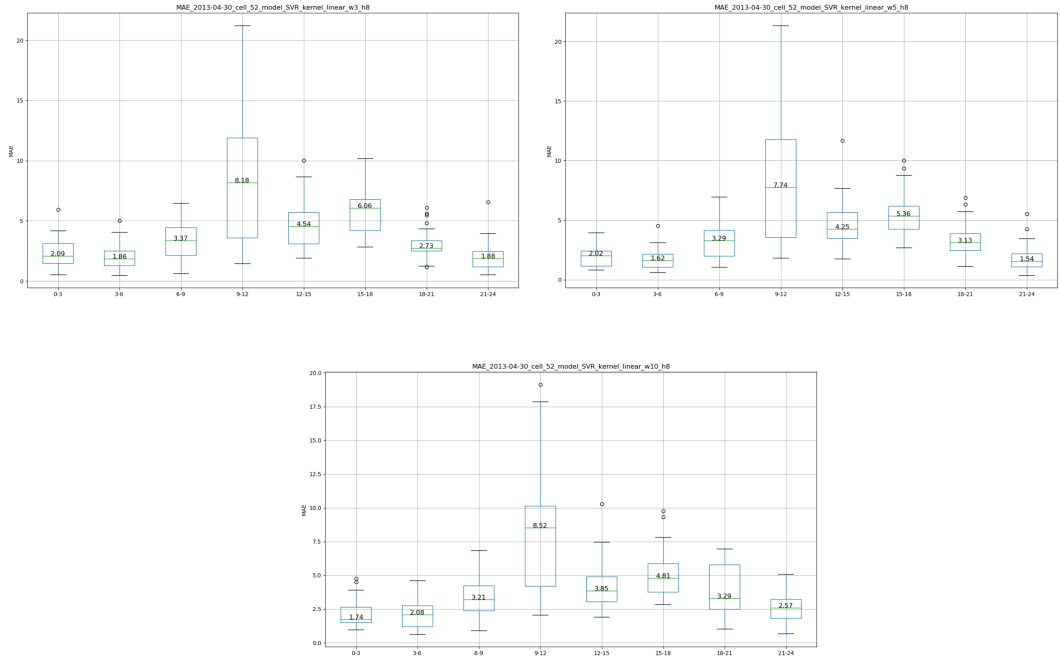


Figura 5.21: Algoritmo SVRlinear, cella 52, data 2013-04-30 e window 3, 5 e 10

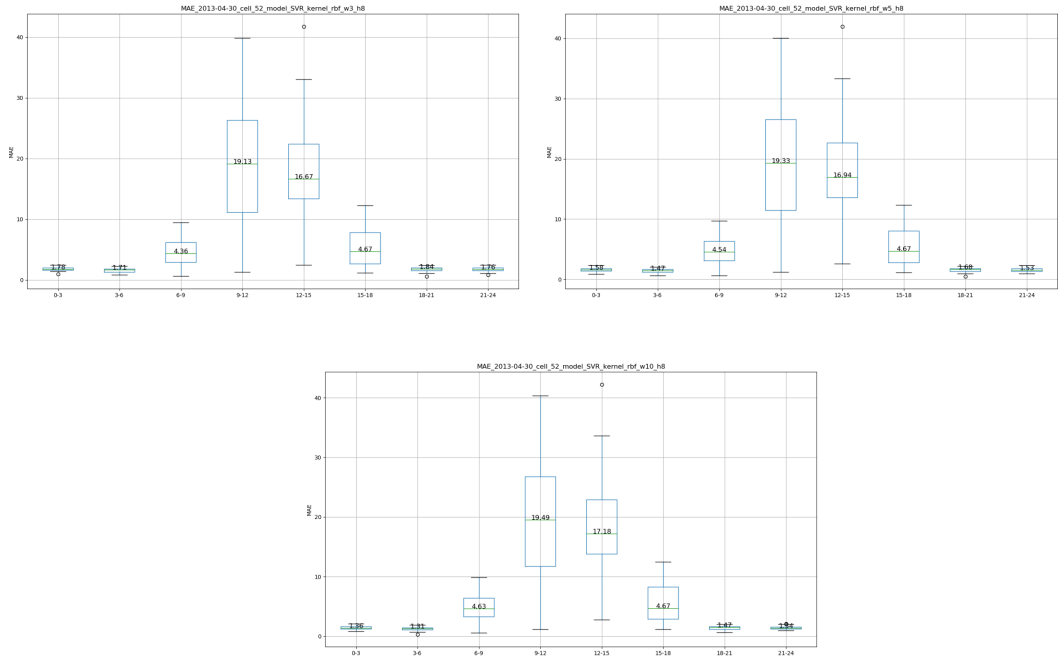


Figura 5.22: Algoritmo SVRrbf, cella 52, data 2013-04-30 e window 3, 5 e 10

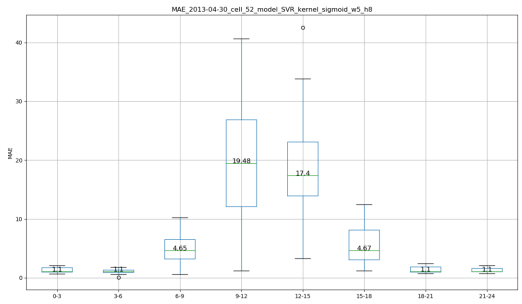


Figura 5.23: Algoritmo SVRsigmoid, cella 52, data 2013-04-30

A questo punto è interessante confrontare l'algoritmo che è stato identificato come meglio performante, ossia RF, con la baseline.
La predizione di baseline non muta in relazione al valore di window, quindi si osserveranno le differenze per RF con window impostata a 3. Per questo confronto sui grafici in figura 5.24 sono evidenziati i valori delle medie. Nella tabella 5.25

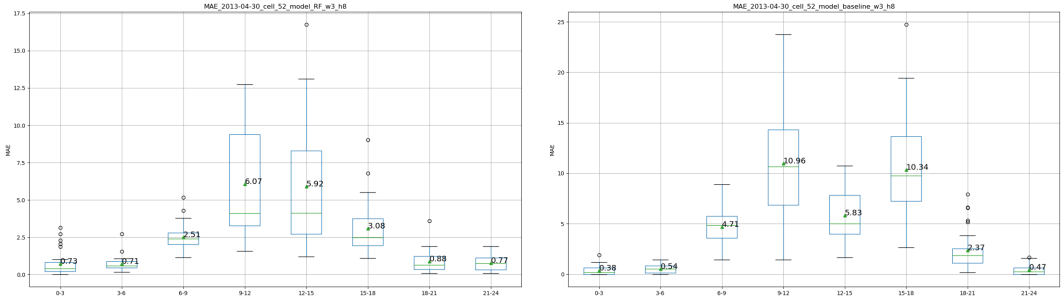


Figura 5.24: Cella 52, data 2013-04-30, a sinistra l'algoritmo RF con window=3, a destra baseline

sono raccolti i valori delle medie di ogni timeslot.
Si osservi che le predizioni per i timeslot corrispondenti agli orari notturni sono migliori di qualche decimo per baseline. Per i timeslot delle ore diurne invece, RF predice nettamente meglio di baseline, eccetto per il timeslot 12-15 dove la differenza tra gli errori di predizione è quasi nulla.

	0-3	**3-6**	**6-9**	**9-12**	**12-15**	**15-18**	**18-21**	**21-24**
RF	0,73	0,71	2,51	6,07	5,92	3,08	0,88	0,77
baseline	0,38	0,54	4,71	10,96	5,83	10,34	2,37	0,47

Figura 5.25: Confronto tra i valori medi di errore per ogni timeslot tra RF e baseline

5.3 Ottimizzazione dell'algoritmo Random Forest

Dagli esperimenti precedenti si è appreso chiaramente che l'algoritmo RandomForestRegressor produce le predizioni migliori rispetto a tutti gli altri algoritmi. Per questo motivo è stata svolta un'analisi focalizzata solamente su questo algoritmo, esplorando le varie combinazioni di parametri e confrontandone le prestazioni.
Le combinazioni sperimentate sulla cella 52 con data 2013-04-30 sono le seguenti, tutte con window pari a 3:

1. versione 'assoluta', R1 R2 R3, feature selection non attiva
2. versione 'assoluta', R1 R2 R3, feature selection attiva
3. versione 'assoluta', R1 R2, feature selection non attiva
4. versione 'assoluta', R1 R2, feature selection attiva
5. versione 'relativa', R1 R2, feature selection non attiva
6. versione 'relativa', R1 R2, feature selection attiva
7. versione 'relativa', R1 R2 R3, feature selection attiva
8. versione 'relativa', R1, feature selection attiva

Nella tabella in figura 5.26 sono raccolti gli errori medi per timeslot e l'errore globale medio, con i valori migliori evidenziati. L'analisi per timeslot permette di eviden-

	0-3	3-6	6-9	9-12	12-15	15-18	18-21	21-24	globale
1	0,73	0,71	2,51	6,07	5,92	3,08	0,88	0,77	2,58375
2	0,8	0,8	2,92	5,96	5,78	4,24	0,93	0,71	2,7675
3	0,77	0,73	2,39	5,84	5,92	3,05	0,86	0,79	2,54375
4	0,85	0,77	2,74	6,08	6,43	4,14	0,99	0,77	2,84625
5	0,41	0,64	2,46	4,91	4,3	5,38	2,19	0,64	2,61625
6	0,4	0,63	2,5	5,05	4,3	5,3	2,11	0,6	2,61125
7	0,4	0,62	2,55	5,27	4,32	5,47	2,12	0,59	2,6675
8	0,42	0,59	2,56	5,25	4,22	5,44	2,1	0,6	2,6475

Figura 5.26: Confronto per varie combinazioni di prarametri dell'algoritmo RF

ziare che, nonostante la performance globale migliore sia prodotta dal modello in versione 'assoluta', il modello in versione 'relativa' performa nettamente meglio nei timeslot più critici per la predizione. Per il timeslot '9-12' la versione 'relativa' raggiunge un errore medio pari a 4,91, quando la versione 'assoluta' raggiunge solamente il valore 5,84. La stessa osservazione vale per il timeslot '12-15' in cui la

versione ‘relativa‘ riesce, nella condizione migliore, a raggiungere un errore medio di 4,22 contro 5,78 della versione ‘assoluta‘.

5.3.1 Influenza del training set

In questa parte di lavoro si vuole indagare se l’aumento della lunghezza del training set porti miglioramenti alla predizione effettuata con RF, sia nella versione ‘assoluta‘ che in quella ‘relativa‘.
Le combinazioni di parametri scelte per il processo predittivo corrispondono alle migliori possibili identificate al passo precedente, la numero 3 per la versione ‘assoluta‘ e la numero 6 per la versione ‘relativa‘.
Il parametro della data è stato variato in accordo con la lunghezza del training set, in modo da eseguire le predizioni sempre per gli stessi giorni.
L’incremento della lunghezza del training set è stata osservata ad aggiunte di 15 giorni per volta.
Gli errori ottenuti con la versione ‘assoluta‘ sono riportati nella tabella in alto della figura 5.27, quelli per la versione ‘relativa‘ in quella in basso.

	0-3	3-6	6-9	9-12	12-15	15-18	18-21	21-24	globale
30 giorni	0,52	0,63	2,4	5,27	5,88	2,87	1,04	0,98	2,44875
45 giorni	0,41	0,57	2,28	5,07	5,52	2,81	0,95	0,99	2,325
60 giorni	0,43	0,57	2,26	5,01	4,62	2,92	0,77	0,83	2,17625

	0-3	3-6	6-9	9-12	12-15	15-18	18-21	21-24	globale
30 giorni	0,36	0,51	2,46	4,47	4,42	4,99	1,96	0,6	2,47125
45 giorni	0,35	0,53	2,3	4,87	4,56	4,97	1,92	0,59	2,51125

Figura 5.27: Prestazioni dell’algoritmo RF per timeslot al variare della lunghezza del training set, versione relativa in basso

Si apprende dagli esperimenti che per la versione ‘relativa‘ definire un training set superiore a 45 giorni non provoca miglioramenti nel processo predittivo. Per la versione ‘assoluta‘ invece si notano miglioramenti fino ad una lunghezza del training set di 60 giorni.
RF nella versione ‘assoluta‘ con un training set lungo 60 giorni risulta essere la configurazione migliore.
Avendo raggiunto un livello di performance maggiore grazie al cambiamento della lunghezza del training set, si ripete il confronto tra RF nella sua migliore configurazione e la baseline.
Gli errori sono visualizzati nella tabella 5.28. Ora RF predice meglio di baseline in

	0-3	3-6	6-9	9-12	12-15	15-18	18-21	21-24	globale
RF	0,43	0,57	2,26	5,01	4,62	2,92	0,77	0,83	2,17625
baseline	0,38	0,54	4,71	10,96	5,83	10,34	2,37	0,47	4,45

Figura 5.28: Confronto tra i valori medi di errore per ogni timeslot tra RF e baseline

tutti i timeslot diurni, in alcuni casi anche con errori medi pari alla metà o meno di quelli di baseline. Baseline nei timeslot notturni realizza degli errori medi di qualche decimo minori rispetto a RF.

Capitolo 6

Conclusioni e sviluppi futuri

Durante la prima parte di analisi dell'utilizzo del servizio sono stati individuati i pattern di utilizzo del sistema di carsharing. Dall'analisi settimanale e quotidiana, sono state individuate delle ciclicità di utilizzo giornaliero, che hanno permesso di etichettare le celle come 'commerciali' o 'residenziali', le prime caratterizzate da una disponibilità di automobili alta durante le ore diurne e molto bassa per le ore notturne e le altre caratterizzate da un andamento complementare a quello appena descritto ma più attenuato.
Questa ciclicità è ben definita durante i giorni infrasettimanali mentre nel finesettimana perde di intensità. Per le celle classificate come 'commerciali' è stato riscontrato un minor uso del servizio durante il periodo invernale rispetto al periodo estivo ed anche un minor uso durante i giorni festivi rispetto ad i giorni feriali.
Sono state anche individuate celle con un'utilizzo del servizio molto scarso, tempo dopo questa rilevazione si è appreso che lo stesso operatore car2go restrinse l'area di erogazione del servizio due anni dopo la rilevazione dei dati utilizzati in questa tesi, probabilmente nella fase di prototipazione non previdero che queste zone sarebbero rimaste poco utilizzate.

Dagli esperimenti di predizione è emerso senza alcun dubbio che l'algoritmo più adatto al modello elaborato per questo problema è il 'Random Forest Regressor', con questo algoritmo sono state generate le predizioni con l'errore più basso.
Per quanto riguarda la modellazione del problema, l'inclusione dei dati riguardanti il meteo non ha apportato alla predizione un miglioramento utile, come nel caso degli algoritmi 'Gradient Boosting Regression Trees' e 'Random Forest Regressor' o nel caso dell'algoritmo 'SVR', per il quale ha generato perfino un degradamento della performance.
Nella scelta del parametro window, nel caso dell'algoritmo 'Random Forest Regressor', il valore 3 è risultato essere il valore migliore. Per gli algoritmi 'SVR', 'Logistic Regression' e 'Lasso' invece il valore di window migliore è risultato essere il 10.
Riguardo alle due versioni, ad un prima analisi è sembrato che la versione 'relativa'

non fosse efficace. In realtà si è scoperto attraverso l'analisi per timeslot che la versione 'relativa', in confronto a quella 'assoluta', predice molto meglio nei timeslot '9-12' e '12-15', ma peggio nei timeslot '15-18' e '18-21', inoltre si è notato che la modellazione del problema 'relativa' è più affine alla predizione mediante l'algoritmo 'SVR' con kernel non lineari.
A proposito del contributo dei dati relativi all'occupazione nelle corone circolari, gli esperimenti hanno rivelato che la loro influenza sull'errore di predizione è poco incidente, durante lo studio mirato sull'algoritmo Random Forest si è notato che non considerare la terza corona circolare porta dei miglioramenti nella predizione. Questo parametro potrà sicuramente essere oggetto di ulteriore analisi in un lavoro successivo, ricercando valori dei raggi delle corone ottimali.
La ricerca della giusta lunghezza del training set, che inizialmente era stata impostata troppo corta, si è rivelata fondamentale nell'ottimizzazione dell'algoritmo Random Forest.
In conclusione, il framework sviluppato ha permesso di testare e valutare con efficacia gli algoritmi e i modelli predittivi teorizzati, permettendo di individuare la combinazione migliore di parametri per il caso studiato.
Un'immediata estensione di questo lavoro potrebbe prevedere lo studio del comportamento di altri algoritmi di predizione, come per esempio le reti neurali. Potrebbe anche essere interessante applicare il modello ad altri casi di studio, così da verificare empiricamente se davvero le condizioni meteorologiche non influenzano l'utilizzo dei sistemi di carsharing. Un altro approccio radicalmente differente al problema potrebbe considerare l'utilizzo delle informazioni sui 'point of interest' per modellare l'area di utilizzo del carsharing come un insieme di poligoni.

Bibliografia

[1] Shaheen, S., Chan, N., Bansal, A., and Cohen, A. Shared Mobility a Sustainablity and Technology Workshop: Definition, Industry Development and Early Understanding. University of California Berkeley Transportation Sustainability Research Center 2015 page 30

[2] Willing, C., Brandt, T. and Neumann, D. Business and Information Systems Engineering, Intermodal Mobility 2017 59:173.

[3] Martin, C. J. The sharing economy: A pathway to sustainability or a nightmarish form of neoliberal capitalism? Ecological Economics, 2016, 121:149–159.

[4] Francesco Ferrero, Guido Perboli, Mariangela Rosano, Andrea Vesco, Carsharing services: An annotated review, Sustainable Cities and Society, Volume 37, 2018, Pages 501-518.

[5] Shaheen, Susan A. ; Martin, Elliot W. ; Cohen, Adam P. ; Chan, Nelson D. ; Pogodzinsk, Mike, Public bikesharing in North America during a period of rapid expansion: understanding business models, industry trends and user impacts, mti report 12-29, 2014, Pages 2-3.

[6] Shaheen, S., Sperling, D., and Wagner, C. (1998). Carsharing in Europe and North America: Past, Present, and Future. Transportation Quarterly, 52(3):35–52.

[7] Shaheen, S. and Cohen, A. Carsharing and Personal Vehicle Services: Worldwide Market Developments and Emerging Trends. (2012). International Journal of Sustainable Transportation, 7(1):5–34.

[8] Schmöller, S., Weikl, S., Müller, J., and Bogenberger, K. (2015). Empirical analysis of free-floating carsharing usage: The munich and berlin case. Transportation Research Part C: Emerging Technologies.

[9] Kopp, J., Gerike, R., and Axhausen, K. W. (2015). Do sharing people behave differently ? An empirical of free-floating car-sharing members. Transportation, pages 449–469.

[10] Sprei, F. and Ginnebaugh, D. Can car sharing facilitate a more sustainable car purchase? Eceee Summer Study, First Fuel Now 2015

[11] Habibi, S., Sprei, F., Englund, C. Comparison of free-floating car sharing services in cities Eceee Summer Study 2017

[12] Firnkorn Jörg and Müller Martin, 2011. "What will be the environmental effects of new free-floating car-sharing systems? The case of car2go in Ulm", Ecological Economics, Elsevier, vol. 70(8), pages 1519-1528.
[13] Prieto, M., Baltas, G., and Stan, V. (2017). Car sharing adoption intention in urban areas: What are the key sociodemographic drivers? Transportation Research Part A: Policy and Practice, 101:218–227
[14] Glotz-Richter, Michael. (2012). Car-Sharing – "Car-on-call" for reclaiming street space. Procedia - Social and Behavioral Sciences. 48. 1454-1463.
[15] Kumar, V. P. and Bierlaire, M. (2012). Optimizing Locations for a Vehicle Sharing System. pages 1–30.
[16] Jorge, D., Molnar, G., Homem, G., and Correia, D. A. (2015). Trip pricing of oneway station-based carsharing networks with zone and time of day price variations. Transportation Research Part B, 81:461–482.
[17] Stefan Schmöller, Klaus Bogenberger, Analyzing External Factors on the Spatial and Temporal Demand of Car Sharing Systems, Procedia - Social and Behavioral Sciences, Volume 111, 2014, Pages 8-17
[18] Henrik Becker, Francesco Ciari, Kay W. Axhausen, Modeling free-floating car-sharing use in Switzerland: A spatial regression and conditional logit approach, Transportation Research Part C: Emerging Technologies, Volume 81, 2017, Pages 286-299
[19] Yang Li, Dimitrios Gunopulos, Cewu Lu, Leonidas Guibas, Urban Travel Time Prediction using a Small Number of GPS Floating Cars International Conference on Advances in Geographic Information Systems 2017
[20] Johannes Müller, Klaus Bogenberger, Time Series Analysis of Booking Data of a Free-Floating Carsharing System in Berlin, Transportation Research Procedia, Volume 10, 2015, Pages 345-354
[21] David Banister, Karen Anderton, David Bonilla, Moshe Givoni and Tim Schwanen Transportation and the Environment Annual Review of Environment and Resources 2011 36:1, 247-270.
[22] J. Xu and J. S. Lim, A new Evolutionary Neural Network for forecasting net flow of a car sharing system, IEEE Congress on Evolutionary Computation, 2007, pp. 1670-1676
[23] https://aaronparecki.com/car2go
[24] https://www.kaggle.com/selfishgene/historical-hourly-weather-data
[25] https://extract.bbbike.org/
[26] https://planet.openstreetmap.org/
[27] https://astronomerrdiff.wordpress.com/2015/07/31/mapping-car2go-portland/
[28] https://www.oregonlive.com/commuting/2015/07/car2go_cuts_st_johns_east_port.html
[29] https://it.wikipedia.org/wiki/Java_(linguaggio_di_programmazione)
[30] https://it.wikipedia.org/wiki/Python

[31] http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html