Previsione della domanda di bike sharing a Seoul attraverso tecniche di machine learning

Abstract

Il bike sharing (noto anche come bicycle-sharing system o bike share program) è una modalità di trasporto in cui delle biciclette vengono rese disponibili al pubblico, in condivisione, per utilizzo individuale di breve periodo. La maggior parte dei sistemi di bike sharing permette di noleggiare e ritirare la bicicletta da una delle stazioni disponibili (dette "dock"), per poi restituirla a un qualsiasi altro dock presente nella città. Risultano evidenti i vantaggi in termini di scorrevolezza del traffico, inquinamento acustico e soprattutto di impatto ambientale, i quali hanno fatto sì che il sistema venisse accolto in molte città su scala mondiale. La recente crescita dell'utenza ha fatto sorgere l'esigenza di sistemi di previsione del numero orario di noleggi, i quali possono aiutare a ridurre i tempi d'attesa e rendere il servizio più efficiente e funzionale.

In questa sede proponiamo alcuni modelli di machine learning (*Linear Regression Model*, *Random Forest*, *Support Vector Machine*, *Multilayer Perceptron* e *Gradient Boosting Machine*) utili a questo scopo. Vengono infine evidenziati i modelli migliori. Le analisi effettuate sono relative alla città metropolitana di Seoul, Corea del Sud.

1. Descrizione del dataset

Il dataset oggetto dell'analisi è costituito da 8760 osservazioni, relative al periodo compreso tra 1/12/2017 e 30/11/2018, e 14 *features* di cui di seguito riportiamo la descrizione:

- *Date*: data in formato year/month/day
- ***Rented_bike_count***: variabile risposta – numero di biciclette noleggiate in una certa ora del giorno (discreta)
- *Hour*: ora del giorno, indicata come 0, 1, …, 23 (discreta)
- *Temperature* (°C): temperatura in °C (continua)
- *Humidity* (%): percentuale di umidità (discreta)
- *Wind_speed* (m/s): velocità del vento (continua)
- *Visibility* (10m): visibilità misurata ogni dieci metri (discreta)
- *Dew_point_temperature* (°C): temperatura di rugiada (continua)
- *Solar_radiation* (MJ/m2): energia solare per metro quadrato (continua)
- *Rainfall* (mm): millimetri di pioggia caduti (continua)
- *Snowfall* (cm): centimetri di neve caduti (continua)
- *Seasons*: stagione (Winter, Spring, Sumer, Autumn)
- *Holiday*: giorno festivo (dicotomica);
- *Functioning_day*: giorno in cui il servizio di bike sharing era attivo (dicotomica).

Durante l'analisi esplorativa abbiamo osservato la distribuzione di frequenza della variabile risposta (Figura 1):

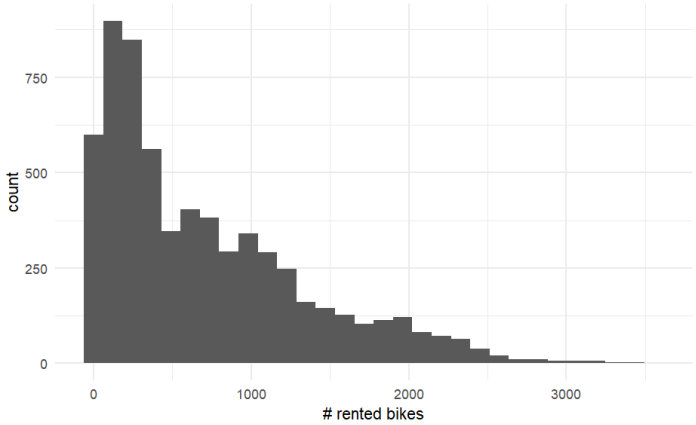

Figura 1: distribuzione del numero di biciclette noleggiate

I valori della variabile target sono compresi in un range tra 0 e 3556. Sono più frequenti i valori attorno a 200. Notiamo inoltre una decrescita simile a un trend esponenziale.

Abbiamo poi studiato la variabile risposta in relazione all'orario e al periodo dell'anno. Si può osservare che la domanda media di biciclette segue un andamento orario analogo tra le quattro stagioni (Figura 2). In particolare, osserviamo un picco a inizio mattina, attorno alle 8:00, presumibilmente per l'affluenza di lavoratori; il secondo picco si osserva alle 18:00, ed è probabilmente dovuto ai lavoratori di rientro a casa e alle persone che noleggiano biciclette per svolgere attività ricreative nel proprio tempo libero.

Osserviamo, inoltre, che la domanda media in primavera, estate e autunno presenta valori simili per la maggior parte della giornata (tra le 5:00 e le 16:00), mentre nelle ore serali si nota un maggiore discostamento per la stagione estiva. La stagione invernale rimane consistentemente quella con la domanda più bassa rispetto a tutte le altre.

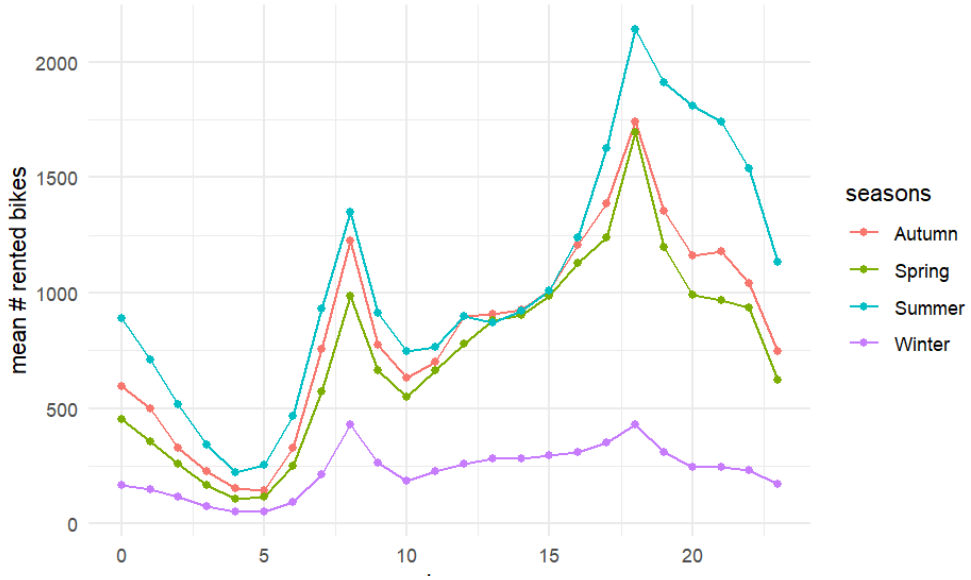

Figura 2: numero medio di biciclette noleggiate a ogni ora, scorporato per stagione

Scorporado ulteriormente le osservazioni, questa volta per mese (Figura 3), notiamo che il picco di domanda si verifica nel mese di giugno. La domanda nei mesi invernali si mantiene abbastanza invariata, rispecchiando l'andamento generale della stagione visto nel grafico precedente.

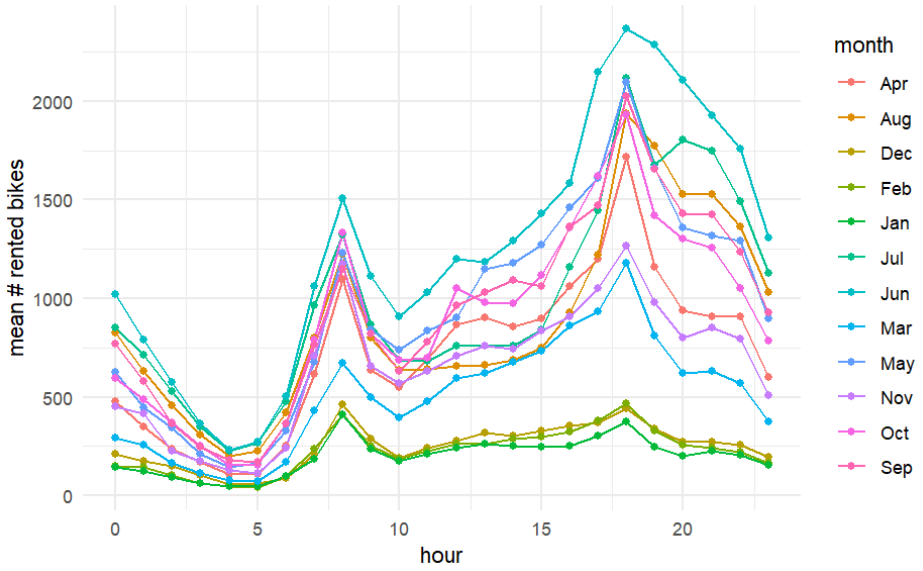

Figura 3: numero medio di bicilette affittate ogni ora, scorporato per mese

Esaminando la distribuzione della variabile risposta rispetto alla temperatura (Figura 4), notiamo che la domanda maggiore si manifesta quando la temperatura si aggira attorno ai 25 °C e ai 30 °C, ragionevolmente in corrispondenza dei mesi estivi. Tra autunno e primavera osserviamo una distribuzione equiparabile, essendo la temperatura nello stesso range (tra 5°C e 20°C). Escludendo i picchi sistematici, dovuti probabilmente a un effetto weekend, notiamo che la distribuzione del numero di biciclette è piuttosto uniforme.

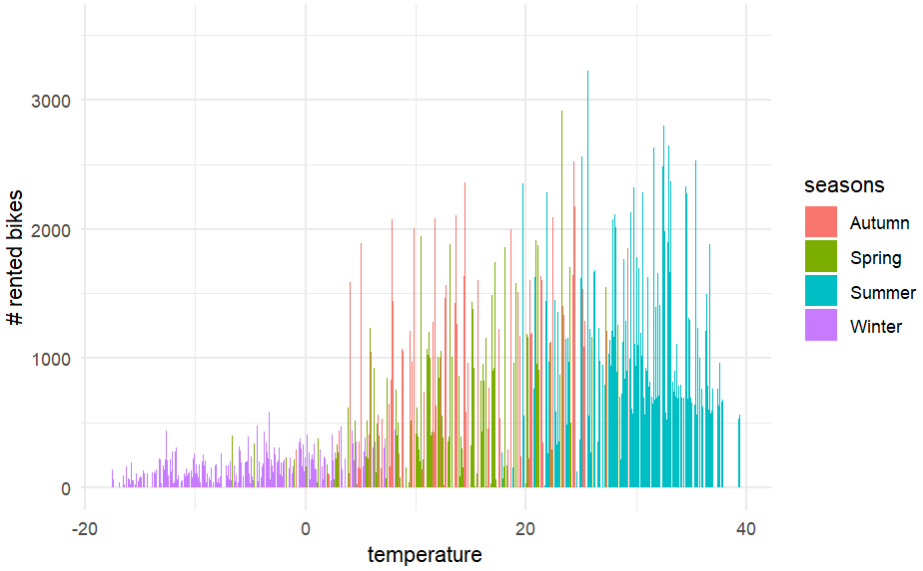

Figura 4: distribuzione del numero di biciclette noleggiate rispetto alla temperatura

Infine, abbiamo studiato la correlazione tra tutte le variabili numeriche (non categoriche) disponibili (Figura 5).

Non risaltano particolari correlazioni, ad eccezione delle seguenti:

- *temperature – dew_point_temperature* (0.91): risultato atteso poiché sono entrambe misure di temperatura;
- *humidity - visibility* (-0.54): correlazione probabilmente dovuta al formarsi della nebbia con l'aumentare dell'umidità;
- *dew_point_temperature - humidity* (0.53): risultato non inaspettato, in quanto per la formazione di rugiada è necessario che sia presente umidità nell'aria;
- *temperature – rented bike count* (0.54): la relazione è stata spiegata nel grafico precedente.

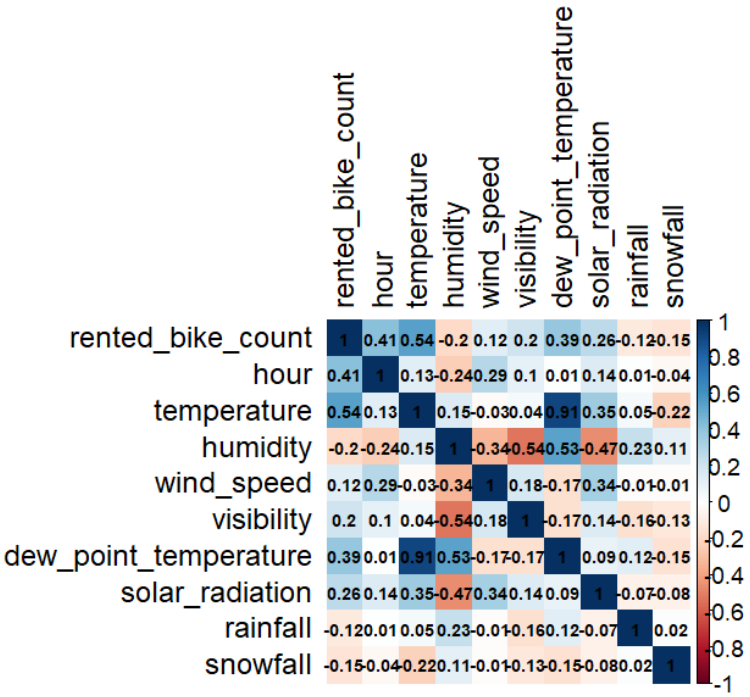

Figura 5: correlazioni tra variabili non categoriche

2. Metodologia

Il dataset originale è stato suddiviso in tre parti, descritte in Tabella 1:

Set	*N° osservazioni*
Training	6307
Validation	1577
Test	876

Tabella 1: suddivisione dataset

Abbiamo utilizzato il *training set* per svolgere le analisi esplorative di cui sopra, e per allenare i modelli previsivi che verranno descritti nei paragrafi successivi.

Tutti i modelli sono stati stimati sul training set con variabili dummy (vedi sezione 2.1). Per ciascuno di essi abbiamo effettuato una stima prima considerando tutte le features disponibili, e poi solo le variabili individuate attraverso la *feature selection* (vedi sezione 2.1), in modo da cogliere variazioni nelle performance previsive.

L'errore empirico e la stima dell'errore di generalizzazione sono stati calcolati con il Root Mean Squared Error (RMSE), così come l'errore di generalizzazione sul test set. In particolare, è stato calcolato prima arrotondando a numeri interi i valori previsti, e poi utilizzando la formula:

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

dove y_i è il valore effettivo della variabile risposta, $\hat{y}_i$ è il valore previsto, n è la dimensione campionaria.

Con il validation set è stata ottenuta una prima stima dell'errore di generalizzazione, eventualmente resa più affidabile da una procedura di cross-validation in caso di valori osservati che potevano far sospettare overfitting da parte del modello. Eventuali tentativi di miglioramento sono stati svolti esaminando il valore dell'RMSE sul training e sul validation set.

Dove possibile, i modelli principali sono stati sottoposti a ricerca degli iperparametri secondo l'approccio *grid search* (il modello viene stimato considerando, una ad una, combinazioni deterministiche dei suoi iperparametri, i cui valori sono scelti a priori dal ricercatore). I nomi degli iperparametri, presenti nelle tabelle relative ai valori ottenuti, fanno riferimento alle implementazioni impiegate.

Infine, dal test set è stato ottenuto l'errore di generalizzazione. Sottolineiamo che è stato impiegato **soltanto al termine** della fase di stima dei modelli, considerandolo come **non disponibile** in tutte le fasi precedenti, evitando così un potenziale problema di data leakage [1].

2.1 Feature selection and engineering

Dal momento che non era di interesse rilevante considerare il dataset come una *serie storica*, si è deciso di rimuovere la variabile *Date* in modo tale da ottenere dei dati in forma *cross-section*. Tuttavia, prima di rimuovere tale variabile è stata sfruttata la sua natura informativa creando, a partire da essa, alcune nuove variabili di possibile interesse per le analisi successive:

- *Weekday:* indica il giorno della settimana a cui fa riferimento l'osservazione (categorica);
- *Weekend:* indica se il giorno a cui fa riferimento l'osservazione è un giorno del weekend (venerdì, sabato o domenica) (dicotomica);
- *Month:* indica il mese a cui fa riferimento l'osservazione (categorica).

Ai fini della stima dei vari modelli abbiamo creato una versione del dataset in cui le variabili categoriche sono espresse come variabili dummy (assumono valore 0 quando non si verifica l'evento di interesse, 1 altrimenti).

[1] Errore di progettazione tale per cui avviene uno scambio di informazioni tra il training e il test set. In fase di training vengono sfruttate informazioni che in realtà sarebbero da considerarsi ignote, portando a performance falsamente elevate al momento della previsione sui dati di test. Nel contesto di questa ricerca, si è evitato di osservare la performance sul test set come linea guida per migliorare i modelli.

Abbiamo cercato di identificare da subito le variabili più rilevanti adattando una *random* forest, utilizzando gli iperparametri di default nell'implementazione dell'algoritmo.

Il criterio con cui sono ordinate le variabili è l'impurità del nodo, calcolata come varianza della variabile risposta nel nodo:

$$\frac{1}{n_{R_j}} \sum_{i \in R_j} \left(y_i - \bar{y}_{R_j} \right)^2$$

dove R_j è la j-esima regione di partizione, generata da una delle variabili esplicative, n_{R_j} è il numero di osservazioni in R_j, y_i è il valore effettivo della variabile risposta osservata nella regione R_j, $\bar{y}_{R_j}$ è la media della variabile risposta nella j-esima partizione. Questa statistica viene calcolata prima su ciascun albero dell'ensemble separatamente, e poi combinata con una media aritmetica [2].

Nella Figura 6 vediamo che le variabili più importanti secondo il criterio appena esposto sono legate alla dimensione temporale (*hour* e *month*), alla temperatura (*temperature* e *solar radiation*) e alle precipitazioni (*humidity* e *rainfall*).

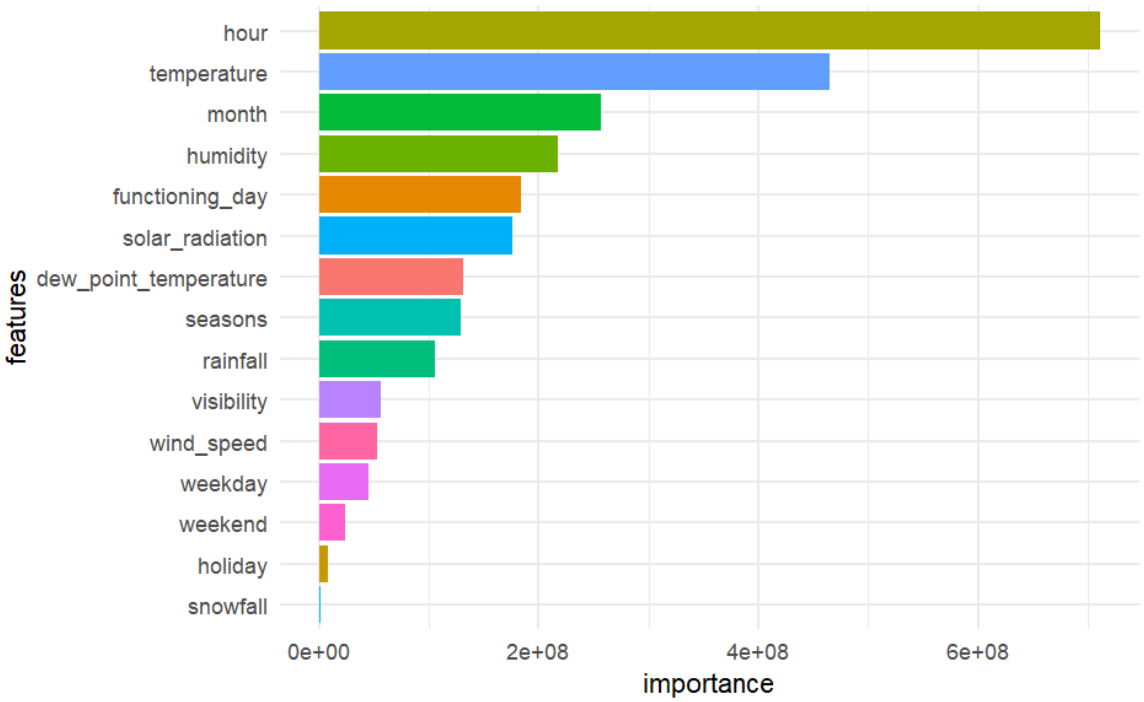

Figura 6: importance plot

[2] *An Introduction to Statistical Learning*: *with applications in R* (Gareth et al.), seconda edizione, pp. 343.

Abbiamo riprodotto il grafico sopra riportato anche per la versione del dataset con variabili dummy (Figura 7), senza ottenere particolari guadagni informativi.

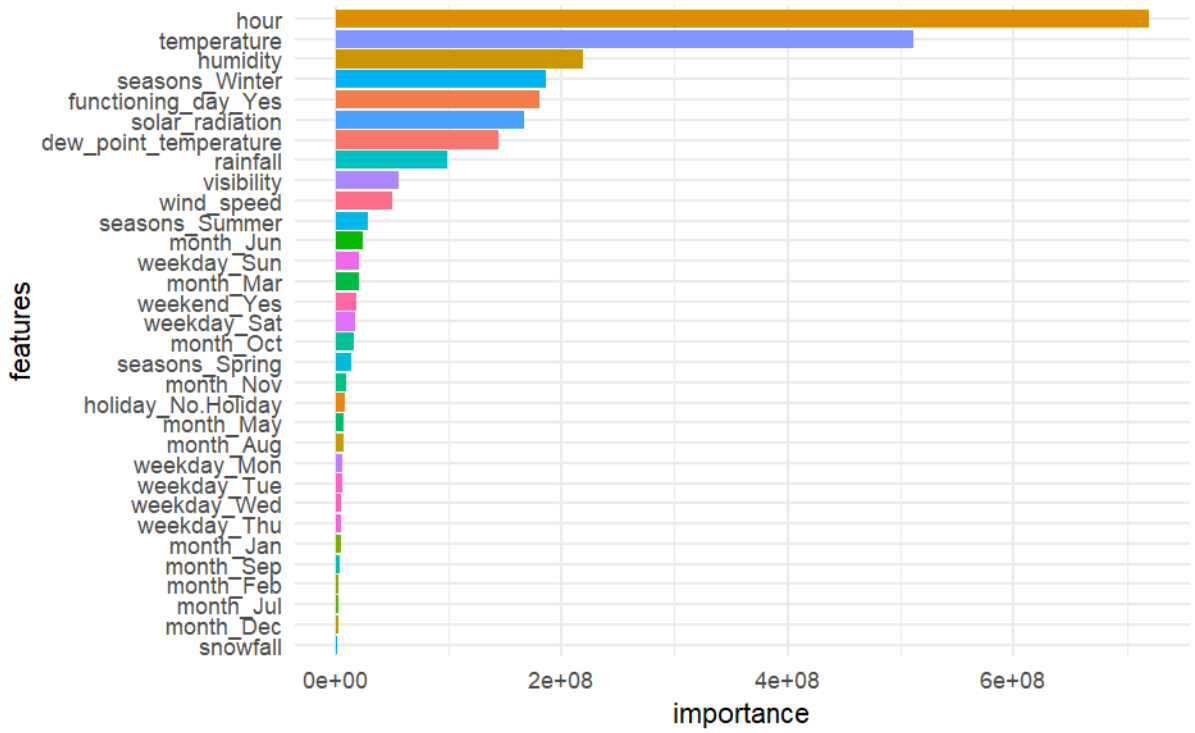

Figura 7: importance plot su dataset con variabili dummy

La *feature selection* è stata effettuata sul dataset con variabili dummy sulla base del grafico appena mostrato. In particolare, abbiamo selezionato le variabili più importanti da *hour* fino a *rainfall*, dopo la quale si osserva un significativo calo di importanza. Di seguito riportiamo le variabili considerate:

hour, temperature, humidity, functioning_day_Yes, seasons_Winter, dew_point_temperature, solar_radiation, rainfall.

3. Modelli

In questa sezione vengono esposti i modelli utilizzati per svolgere il task di interesse. Presentiamo prima i modelli stimati utilizzando tutte le variabili ("all features"), poi i risultati dei modelli stimati con la feature selection ("selected features"), e infine gli esiti dei modelli che abbiamo deciso di migliorare ulteriormente visti i buoni risultati della prima stima ("all features" ottimizzati).

3.1 Modelli "all features"

I modelli illustrati in questo paragrafo sono stati stimati utilizzando tutte le variabili disponibili.

3.1.1 Linear Regression Model (LM)

Modello di regressione lineare, stimato con i minimi quadrati ordinari, e impiegato come benchmark per modelli più sofisticati in grado di valorizzare tutti i tipi di variabili presenti. Per tutti i modelli di regressione lineare stimati abbiamo escluso la variabile *dew_point_temperature*, data l'alta correlazione con la variabile *temperature*, che può alterare la stima del relativo coefficiente.

3.1.2 Random Forest (RF)

La scelta della random forest è stata motivata dalla presenza di variabili sia numeriche sia categoriche, e dalla capacità di questo algoritmo di gestire entrambe in modo efficace anche senza una particolare ottimizzazione degli iperparametri. Il modello stimato utilizzando i valori di default degli iperparametri ha restituito un RMSE sul training set pari a 85.09 e sul validation set pari a 193.32. Un'ulteriore motivazione a sostegno dell'utilizzo di questo modello è la robustezza delle previsioni, in quanto ottenute come media dei risultati di un elevato numero di alberi decisionali.

In Tabella 2 riportiamo gli iperparametri di interesse con relativa descrizione e valore ottimo, ottenuto attraverso la *grid search*.

Iperparametro	*Descrizione*	*Valore ottimo*
`num.trees`	numero di alberi decisionali che compongono la random forest	500
`mtry`	numero di variabili selezionate casualmente a ogni split	12
`min.node.size`	numero minimo di osservazioni nel nodo dopo lo split	1
`replace`	campionamento con o senza reinserimento	FALSE
`sample.fraction`	frazione di osservazioni da campionare (attivo solo in caso di campionamento senza reinserimento)	0.8

Tabella 2: iperparametri random forest

3.1.3 Support Vector Machines (SVM)

Le SVM sono state adattate al dataset riscalato con *standard scaling*: $\frac{X_i - \bar{X}_i}{s_i},\ i = 1, \dots, k$, dove X_i è il vettore della i-esima feature, $\bar{X}_i$ e s_i sono rispettivamente la sua media e deviazione standard. La SVM è sensibile alla scala dei dati: dimensioni con un ampio range di variazione possono influenzare, più di altre, il calcolo dell'iperpiano di separazione.

Il modello stimato utilizzando i valori di default degli iperparametri, la cui implementazione considera un kernel di tipo *radial basis function* (*rbf*), ha restituito un RMSE sul training set pari a 585.58 e pari a 586.40 sul validation set.

Abbiamo successivamente ottimizzato gli iperparametri, considerando come tale anche il kernel. I kernel considerati sono stati: *"lineare", "polinomiale", "rbf"* e *"sigmoide"*.

Il kernel lineare, da cui non ci si aspettava una performance ottimale, è stato preso in considerazione come benchmark per le SVM che sfruttano kernel non lineari. L'RMSE ottenuto con tale kernel è risultato pari a 431.84 sul training set e 436.97 sul validation set.

La performance migliore è stata ottenuta utilizzando il kernel *"rbf"*, con un training RMSE di 318.59 e un validation RMSE di 335.01.

Iperparametro	*Descrizione*
`kernel`	Kernel utilizzato dall'algoritmo (lineare, polinomiale, rbf e sigmoide)
`gamma`	Coefficiente per i kernel rbf, polinomiale e sigmoide
`C`	Parametro di regolarizzazione
`epsilon`	Soglia oltre la quale non viene associata una penalizzazione alla loss function calcolata sul training
`degree`	Grado del polinomio (utilizzato solo nel kernel polinomiale)

Tabella 3: iperparametri SVM

In Tabella 4 vengono riportati i valori ottimi degli iperparametri ottenuti per ciascuno dei kernel considerati (il kernel lineare è assente in quanto non ha iperparametri):

Iperparametro	*Valori ottimi*		
`kernel`	*rbf*	*sigmoide*	*polinomiale*
`gamma`	0.1	0.01	-
`C`	100	100	100
`epsilon`	0.5	0.5	0.1
`degree`	-	-	3

Tabella 4: valori degli iperparametri delle SVM ottenuti dalla grid search

3.1.4 Rete Neurale - Multilayer Perceptron (MLP)

Le reti neurali sono state allenate sui dati riscalati con *standard scaling* (vedi 3.1.3 per la formula), in quanto la presenza di input con valori particolarmente elevati può attivare ampi aggiornamenti del gradiente della loss function, impendendo alla rete di convergere. La loss function utilizzata per il training è l'MSE, ottimizzata attraverso l'algoritmo ADAM.

In Tabella 5 riportiamo le descrizioni degli iperparametri di interesse:

Iperparametro	*Descrizione*
`# hidden layers`	Numero di hidden layers
`# neurons per layer`	Numero di neuroni per layer
`activation`	Funzione di attivazione a ogni layer
`epochs`	Numero di iterazioni
`learning rate`	Tasso di apprendimento
`optimizer`	Algoritmo di ottimizzazione della loss function

Tabella 5: iperparametri MLP

In Tabella 6 viene esplicitata la struttura della rete neurale. Si può notare una disposizione "triangolare": il numero di neuroni è elevato nei primi *layer*, per poi decrescere proporzionalmente a ogni *layer* successivo. Nonostante la scarsa dimensione della rete, è stata osservata una buona performance in termini di RMSE sul training set (34.29) e sul validation set (171.27). Una possibile spiegazione è che questa rete riesce a individuare precisamente il pattern generale dei dati nei primi layer, per poi, nei layer finali, apprendere i dettagli senza un adattamento eccessivo.

	Layer type	*# neurons*	*# weights and biases*	*Activation*
Input layer	-	32 + 1	-	
Layer 1	Dense	256	8’448	ReLu
Layer 2	Dense	128	32’896	ReLu
Layer 3	Dense	32	4’128	ReLu
Output layer	Dense	1	33	Linear

Tabella 6: struttura del MLP

Il discostamento di RMSE tra il training set e il validation set può far sospettare overfitting da parte del modello, abbiamo quindi adottato la strategia di regolarizzazione di tipo *dropout*. Questo tipo di regolarizzazione, applicabile a ogni layer singolarmente, consiste nella disattivazione di una frazione (detta *dropout rate*) di neuroni, scelti casualmente dall’algoritmo a ogni *epoch*. Fare ciò rende i neuroni meno sensibili a variazioni negli input ricevuti, irrobustendo la rete neurale e aumentando la sua capacità di generalizzazione. L’utilizzo del *dropout* ha dato l’esito sperato: è stato ridotto l’overfitting nel training set e mantenuto il risultato sul validation set (vedi Tabella 10).

3.1.5 Gradient Boosting Machine (GBM)

La GBM è un modello basato su un ensemble di alberi decisionali che vengono fatti crescere sequenzialmente. Ciascun albero corregge le previsioni di quello precedente attraverso stime sempre più accurate dei residui, ad una velocità che dipende dal learning rate, il quale controlla la frazione di residuo per cui devono essere corrette le previsioni a ogni passaggio. Di seguito riportiamo una sintesi dei passaggi dell’algoritmo di boosting:

Algoritmo `Gradient Tree Boosting` [3]

0. Imposta learning rate λ e numero di iterazioni B;
1. Poni $\hat{f}(x) = 0$ e $r_i = y_i$ per ogni i nel training set;
2. Per $b = 1, \dots, B$, ripeti:
    a. Adatta un albero decisionale $\hat{f}^b$ al training set (X, r), ovvero effettua la previsione sui residui utilizzando le features disponibili;
    b. Aggiorna $\hat{f}$ sommando una frazione dei residui previsti nel passo a:
    $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$
    c. Aggiorna i residui: $r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$
3. Restituisci il modello “boosted”: $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$

Abbiamo stimato una prima GBM con gli iperparametri di default dell’implementazione utilizzata, ottenendo una performance di RMSE sul training set pari a 230.04 e 251.17 sul validation set.

È stata poi effettuata la *grid search*, ottenendo risultati analoghi alla rete neurale (Tabella 8). Abbiamo poi svolto una seconda ricerca a partire da un intorno dei valori degli iperparametri ottenuti dalla ricerca precedente, migliorando ulteriormente la performance previsiva sul validation set.

[3] Tratto da *An Introduction to Statistical Learning*: *with applications in R* (Gareth et al.), seconda edizione, pp. 347.

Iperparametro	*Descrizione*	*Search 1*	*Search 2*
`max_depth`	Numero massimo di nodi per ciascun albero	6	7
`learning_rate`	Tasso di apprendimento	0.12	0.12
`n_estimators`	Numero di fasi boosting da effettuare	500	800

Tabella 7: iperparametri GBM

Tuttavia, i valori di training RMSE ottenuti in entrambi i casi hanno fatto supporre overfitting. È stata quindi impiegata la regolarizzazione *subsample,* che consiste nel campionare casualmente una porzione delle istanze di training a ogni passaggio di boosting, diminuendo la varianza e aumentando il bias dell'ensemble e velocizzando l'allenamento. Il valore del *subsample* (in entrambi i casi pari a 0.8) è stato anch'esso individuato tramite *grid search*, fissando i valori degli altri iperparametri.

3.1.6 Confronto performance

Di seguito vengono riportati i valori dell'RMSE per ciascuno dei modelli rilevanti stimati utilizzando tutte le variabili. In Tabella 8 vediamo performance simili da parte dei modelli MLP e GBM sul training set, con una performance leggermente migliore (circa 3%) per la GBM sul test set. Si nota che la RF effettua previsioni analoghe a quelle dei due modelli appena menzionati sul training set, ma con un maggiore errore di generalizzazione.

Model	*Train RMSE*	*Valid RMSE*	*Test RMSE*	*Note*
LM	409.52	414.07	416.66	
RF	35.93	182.41	186.78	
SVM	318.59	335.01	344.28	Kernel: rbf
MLP	34.29	171.27	166.53	
GBM	31.72	160.01	161.63	1st Hyperparam. Search

Tabella 8: performance modelli "all features"

3.2 Modelli "selected features"

Abbiamo studiato l'effetto della rimozione delle variabili meno importanti, mantenendo solo quelle riportate nella sezione 2.1, relativa alla *feature selection*.

La procedura di stima è stata eseguita in modo analogo a quanto esposto nella sezione precedente (sezione 3.1). In particolare, gli iperparametri di ciascun modello sono stati ottimizzati attraverso la *grid search*.

Abbiamo osservato un peggioramento di performance previsiva da parte di tutti i modelli, ad eccezione della SVM, che presenta una riduzione dell'errore previsivo su tutti i set. In particolare, si evidenzia una riduzione del 18.6% dell'errore di generalizzazione sul test set.

Model	*Train RMSE*	*Valid RMSE*	*Test RMSE*	*Note*
LM	435.77	437.20	445.99	
RF	115.36	244.52	251.57	
SVM	277.47	286.65	280.07	Kernel: rbf
MLP	221.35	246.74	248.91	
GBM	160.44	243.28	245.88	1st Hyperparam. Search

Tabella 9: performance modelli "selected features"

Riportiamo in Tabella 10 anche la struttura della rete neurale stimata in questo contesto.

	Layer type	*# neurons*	*# weights and biases*	*Activation*
Input layer	-	8 + 1	-	
Layer 1	Dense	64	576	ReLu
Layer 2	Dense	32	2080	ReLu
Layer 3	Dense	8	264	ReLu
Output layer	Dense	1	9	Linear

Tabella 10: struttura del MLP "selected features"

Il numero di neuroni è stato riadattato al numero di features in input, mantenendo le stesse proporzioni della prima rete neurale (quella stimata nel contesto “all features”).

3.3 Modelli “all features” ottimizzati

Di seguito riportiamo le performance dei modelli con la performance complessiva migliore, vale a dire MLP e GBM, ulteriormente elaborati secondo quanto descritto nei relativi paragrafi della sezione 3.2.

Model	*Train RMSE*	*Valid RMSE*	*Test RMSE*	*Notes*
MLP	95.30	164.58	164.44	Dropout
GBM	24.92	159.16	160.96	1st Search + Subsample
GBM	5.48	151.74	162.30	2nd Hyperparam. Search
GBM	3.46	152.28	151.20	2nd Search + Subsample

Tabella 11: performance modelli “all features” ottimizzati

In merito al modello MLP abbiamo osservato un peggioramento del RMSE sul training set che fa pensare a una riduzione del presunto overfitting. Il risultato viene confermato dal test RMSE, leggermente inferiore a quello del modello MLP non regolarizzato.

Per il GBM *first search* regolarizzato con il *subsampling* vediamo risultati simili a quelli ottenuti con il modello non regolarizzato. La GBM *second search* presenta un valore estremamente basso per il training RMSE, ma un simile RMSE sul test set rispetto alla GBM first search. Sorprendentemente, la regolarizzazione applicata alla GBM della seconda ricerca ha dato un training RMSE ancora inferiore, e una performance sul test set significativamente migliore rispetto alla sua versione non regolarizzata.

Dopo che è stato osservato il distacco tra training RMSE e validation RMSE per le GBM qui considerate, abbiamo investigato l’eventuale overfitting tramite una procedura di 10-fold cross-validation sul training set. La Tabella 12 riporta alcune statistiche descrittive relative ai risultati ottenuti:

Model	*RMSE mean*	*RMSE std. dev.*	*Notes*
GBM	151.73	10.51	1st Hyperparam. Search
GBM	154.05	9.42	1st Search + Subsample
GBM	153.07	11.40	2nd Hyperparam. Search
GBM	153.94	9.18	2nd Search + Subsample

Tabella 12: cross-validation RMSE – modelli GBM

Gli esiti della cross-validation ci allontanerebbero dall’ipotesi di overfitting, essendo stati osservati RMSE che variano in un range di valori simili a quelli precedentemente ottenuti sul test set.

4. Conclusioni

Dalle analisi eseguite abbiamo osservato come l'utilizzo di tutte le variabili migliori la performance su tutti i modelli, rispetto a una selezione delle variabili basata sul criterio dell'importanza.

Abbiamo osservato performance ottimali da parte dei modelli più complessi, in particolare da quelli regolarizzati (MLP e GBM) confrontati con le rispetti versioni non regolarizzate. Nonostante il presunto overfitting abbiamo osservato consistentemente valori del RMSE sul test set che confutavano tale ipotesi e si mantenevano su un livello costante, senza particolari variazioni.

Dai risultati ottenuti, possiamo quindi concludere che il modello migliore in assoluto in termini di errore di generalizzazione è la GBM regolarizzata con *subsample*, con gli iperparametri della seconda ricerca. Nonostante l'esito della cross-validation, che farebbe escludere l'overfitting, non riponiamo totale fiducia in tale modello dato il valore estremamente basso, e quindi sospetto, del training RMSE.

Per questo motivo, terremmo in considerazione anche il modello MLP regolarizzato con dropout, il quale ha manifestato un comportamento più stabile tra errore empirico ed errore di generalizzazione.

Bibliografia

- James G., Witten D., Hastie T., Tibshirani R. 2021. *An Introduction to Statistical Learning: with Applications in R* - (2nd ed.). Springer.

Implementazioni utilizzate

Di seguito riportiamo i pacchetti utilizzati per lo splitting dei dati e la stima dei modelli.

R

- Versione di R: 4.2.0 – "Vigorous Calisthenics"
- Splitting dati: `rsample` – versione 0.1.1
- Random forest: `ranger` – versione 0.13.1

Python

- Versione di Python: 3.10.4
- SVM e GBM: `scikit-learn (sklearn)` – versione 1.0.2
- MLP: `keras` – versione 2.8.0