Tecniche di Deep Learning per la Previsione di Consumi Energetici

Relazione Progetto di Ingegneria Informatica 2020

Enrico Grazioli
Matricola 871569

1. Descrizione del Progetto

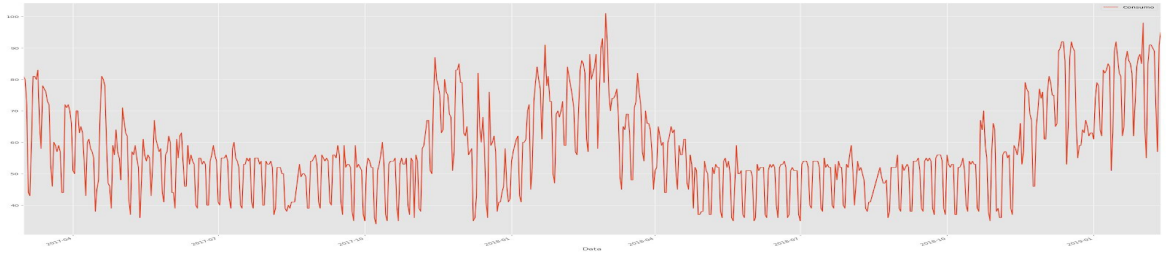
Il progetto descritto in questo documento consiste nell'implementazione di modelli di apprendimento tramite reti neurali di dati riguardanti il consumo energetico di un palazzo situato a Bergamo. Si tratta di due serie di campionamenti, uno a cadenza giornaliera effettuato su un periodo di due anni, l'altro a cadenza oraria effettuato sul mese di Gennaio 2019.
Dopo un iniziale tentativo di approccio prettamente autoregressivo, si è rivelato più proficuo concentrarsi sull'arricchimento dei dati di input del modello tramite la creazione di feature riguardanti le caratteristiche di un determinato giorno dell'anno (feriale, festivo, weekend ecc.) e l'aggiunta di dati meteorologici.

2.Analisi e Completamento dei Dati

Questa fase del progetto consiste in tre operazioni principali. Innanzitutto è stata eseguito l'inserimento dei dati mancanti, così da poter sfruttare l'intero dataset. Poi sono stati scaricati e integrati i dati storici riguardanti le condizioni meteo della città di bergamo nei periodi presi in considerazione. Infine sono state prese in considerazione le caratteristiche specifiche del calendario, individuando weekend e giorni festivi, così come altri dati legati al periodo dell'anno.
È importante specificare che non tutti i dati ricavati in questa fase sono poi effettivamente stati utilizzati nel modello, ma sono stati poi testati e selezionati a seconda dell'efficienza del modello risultante.

2.1.Dati giornalieri

2.1.1.Completamento dei Dati



Per questa operazione non sono state adottate tecniche di machine learning o altre tecniche statistiche poiché la mole di dati mancanti non era tale da giustificarne l'impiego. Infatti questa è la lista completa dei valori mancanti:
['2017-05-14', '2017-08-19', '2017-08-20', '2018-01-21', '2018-05-21', '2018-08-05', '2018-08-15', '2018-08-16', '2018-08-17', '2018-08-18', '2018-08-19']
Sono in tutto 11 e si può notare che sono in gran parte concentrati nel periodo di ferragosto, dove i dati sono caratterizzati da bassa varianza e relativa stabilità. È quindi difficile introdurre errori sistematici e di conseguenza il completamento manuale dei dati è stato considerato il più efficiente.

2.1.2.Inserimento dei Dati Meteo

I dati qui trattati sono stati scaricati dal sito https://www.ilmeteo.it/portale/archivio-meteo/Bergamo. I dati già in questa fase hanno subito un parziale processo di selezione.

TMEDIA C	min: -5.0	max: 30.0
TMIN C	min: -7.0	max: 24.0
TMAX C	min: -3.0	max: 36.0
PUNTORUGIADA C	min: 0.0	max: 20.0
UMIDITA prc	min: 22.0	max: 100.0
VISIBILITA km	min: 1.0	max: 23.0
VENTOMEDIA km/h	min: 3.0	max: 21.0
VENTOMAX km/h	min: 5.0	max: 50.0
RAFFICA km/h	min: 0.0	max: 0.0
PRESSIONESLM mb	min: 992.0	max: 1036.0
PRESSIONEMEDIA mb	min: 0.0	max: 0.0

PIOGGIA mm min: 0.0 max: 0.0
FENOMENI : Attributo categorico.

Si può infatti già notare dal sommario che le voci RAFFICA km/h, PRESSIONEMEDIA mb e PIOGGIA mm hanno minimo e massimo uguale, con l'evidente conseguenza che non possono costituire un dato significativo per il modello.
La voce fenomeni ha dovuto subire un'altra trasformazione, essendo una variabile categorica è stata codificata con il metodo delle dummy variables, ovvero è stato creato un numero di feature binarie pari al numero di categorie.

Data	Pioggia	Neve	Nebbia	Temporale
2019-01-16	1	0	0	0
2019-01-17	1	0	0	0
2019-01-19	1	1	0	0
2019-01-27	1	1	0	0
2019-01-30	0	1	0	0
2019-01-31	1	1	1	0
2019-02-01	1	1	0	0
2019-02-02	1	0	0	0
2019-02-03	1	0	0	0
2019-02-10	1	0	0	0

Data	TMEDIA C	TMIN C	TMAX C	PUNTORUGIADA C	UMIDITA prc	VISIBILITA km	VENTOMEDIA km/h	VENTOMAX km/h	PRESSIONESLM mb
2019-02-03	5.0	3.0	7.0	4.0	96.0	6.0	6.0	11.0	1003.0
2019-02-04	5.0	1.0	11.0	2.0	69.0	20.0	8.0	13.0	1021.0
2019-02-05	6.0	1.0	10.0	1.0	66.0	21.0	7.0	14.0	1020.0
2019-02-06	6.0	2.0	11.0	0.0	63.0	21.0	8.0	13.0	1023.0
2019-02-07	4.0	0.0	8.0	1.0	69.0	15.0	6.0	13.0	1019.0
2019-02-08	5.0	0.0	12.0	0.0	63.0	20.0	7.0	13.0	1019.0
2019-02-09	6.0	1.0	12.0	0.0	59.0	19.0	7.0	17.0	1019.0
2019-02-10	6.0	4.0	7.0	5.0	93.0	5.0	5.0	13.0	1014.0
2019-02-11	7.0	2.0	13.0	3.0	52.0	18.0	12.0	21.0	1011.0
2019-02-12	6.0	1.0	12.0	0.0	46.0	21.0	8.0	13.0	1024.0

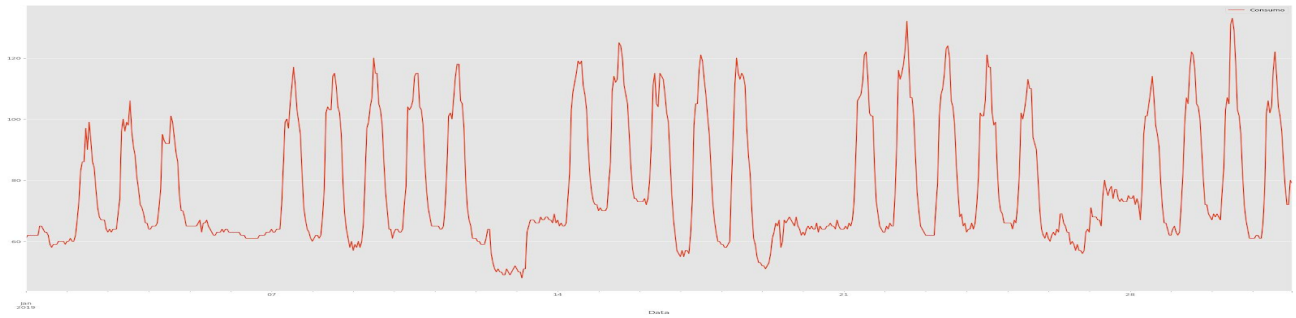
2.1.3. Inserimento Dati da Calendario

Questa fase del completamento dei dati consiste in un tentativo di estrarre dal calendario ogni tipo di informazione che possa influenzare il consumo di energia. Segue una lista delle feature create.

- **Tramonto**/**Alba**: l'ora del tramonto è un evidente fattore che può influenzare i consumi energetici in un certo luogo, oltre che per l'accensione della luce nelle ore di buio, essa può dare al modello un'efficace rappresentazione della stagionalità annuale dei dati. Il discorso è analogo per l'alba. I dati sono espressi in minuti dalla mezzanotte.
- **Feste**: In un edificio di uffici, come sembra essere l'edificio in questione a giudicare dal comportamento dei dati in corrispondenza del weekend, i consumi sono dovrebbero tendere a scendere. Si tratta di una variabile binaria e per l'estrazione delle date di festa è stata utilizzata la libreria holidays.
- **Giorni della settimana**: 7 variabili binarie, una per giorno della settimana, potrebbe catturare comportamenti specifici di alcuni giorni della settimana.
- **Weekend**: variabile binaria che accorpa Sabato e Domenica.
- **Natale**: variabile binaria che identifica il periodo della settimana del 24 Dicembre che non era identificata come periodo di vacanza nella libreria.
- **Ferragosto**: variabile binaria che identifica il periodo della settimana del 14 Agosto che non era identificata come periodo di vacanza nella libreria.
  Entrambe le ultime variabili sono state accorpate alla variabile Feste.

Data	Tramonto	Alba	Feste	Lunedi	Martedi	Mercoledi	Giovedi	Venerdi	Sabato	Domenica	Weekend	Natale	Ferragosto
2017-03-01	1087	420	0	0	0	1	0	0	0	0	0	0	0
2017-03-02	1088	418	0	0	0	0	1	0	0	0	0	0	0
2017-03-03	1089	416	0	0	0	0	0	1	0	0	0	0	0
2017-03-04	1091	414	0	0	0	0	0	0	1	0	1	0	0
2017-03-05	1092	413	0	0	0	0	0	0	0	1	1	0	0
2017-03-06	1094	411	0	1	0	0	0	0	0	0	0	0	0
2017-03-07	1095	409	0	0	1	0	0	0	0	0	0	0	0
2017-03-08	1096	407	0	0	0	1	0	0	0	0	0	0	0
2017-03-09	1098	405	0	0	0	0	1	0	0	0	0	0	0
2017-03-10	1099	403	0	0	0	0	0	1	0	0	0	0	0

2.2.Dati orari



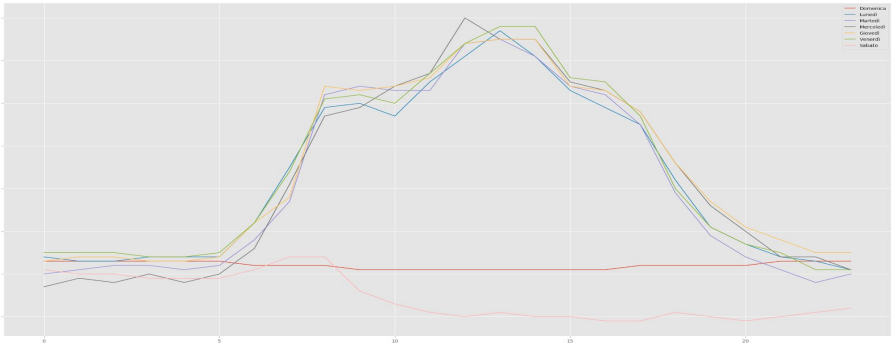
In questo caso i dati erano completi quindi solo le ultime due fasi sono state necessarie.

2.2.1.Inserimento dei Dati Meteo

Questa fase è identica a quella descritta per i dati giornalieri, infatti non sono disponibili dati storici orari per quanto riguarda i parametri meteorologici quindi i dati riguardanti un certo giorno sono stati ripetuti per ogni ora del giorno stesso. Rimando alla Sezione 2.1.2 per dettagli.

2.2.2.Dati da Calendario

Anche qui si è cercato di “etichettare” orari specifici cercando di catturare determinati comportamenti del grafico. Le conclusioni sono state tratte principalmente mettendo a l’andamento dei consumi nei diversi giorni della stessa settimana come di seguito.



Seguono le feature create:

- Ore al Tramonto: a differenza dei dati giornalieri è stato utilizzato un dato differenziale quindi non più l'orario del tramonto nel dato giorno ma i minuti mancanti al tramonto dall'ora attuale.
- Ferie: come nei dati giornalieri ma qui meno significativo poiché solo uno dei giorni considerati è festivo.
- Giorni della Settimana: come per i dati giornalieri.
- Orario Lavorativo: qui sono state etichettare le ore che in Italia si considerano orario di lavoro ovvero quelle comprese tra le 08.00 e le 19.00.
- Mattinata/Pomeriggio/Sera/Notte: qui si è cercato di mettere in evidenza alcune sezioni specifiche della giornata che corrispondono, nei giorni lavorativi, ad una crescita/decrescita/stazionarietà dei consumi.

Data	Ferie	Lunedi	Martedi	Mercoledi	Giovedi	Venerdi	Sabato	Domenica	Orario Lavorativo	Mattinata	Pomeriggio	Serata	Notte
2019-01-01 00:00:00	1	0	1	0	0	0	0	0	0	0	0	1	0
2019-01-01 01:00:00	1	0	1	0	0	0	0	0	0	0	0	0	1
2019-01-01 02:00:00	1	0	1	0	0	0	0	0	0	0	0	0	1
2019-01-01 03:00:00	1	0	1	0	0	0	0	0	0	0	0	0	1
2019-01-01 04:00:00	1	0	1	0	0	0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2019-01-31 19:00:00	0	0	0	0	1	0	0	0	1	0	1	0	0
2019-01-31 20:00:00	0	0	0	0	1	0	0	0	1	0	1	0	0
2019-01-31 21:00:00	0	0	0	0	1	0	0	0	0	0	0	1	0
2019-01-31 22:00:00	0	0	0	0	1	0	0	0	0	0	0	1	0
2019-01-31 23:00:00	0	0	0	0	1	0	0	0	0	0	0	1	0

3. Modelli

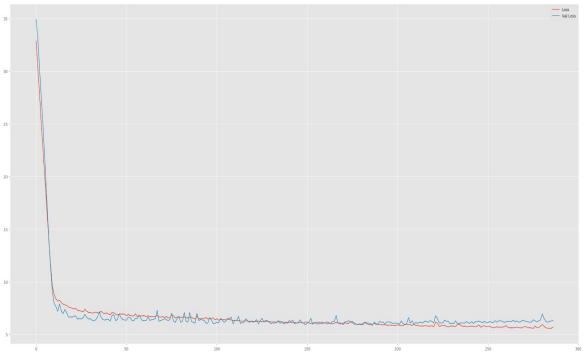
I modelli sviluppati sono degli MLP, l'ottimizzazione è stata attuata testando diverse caratteristiche della rete come funzione di attivazione, numero di strati, neuroni per strato, algoritmo di training. È inoltre stato selezionato per ogni modello il sottoinsieme di feature ottimale. La valutazione dei modelli è stata eseguita secondo il criterio del MAPE.
Nelle sezioni seguenti riporto codice e grafici di training dei modelli.
Per evitare training inutile o problemi di overfitting

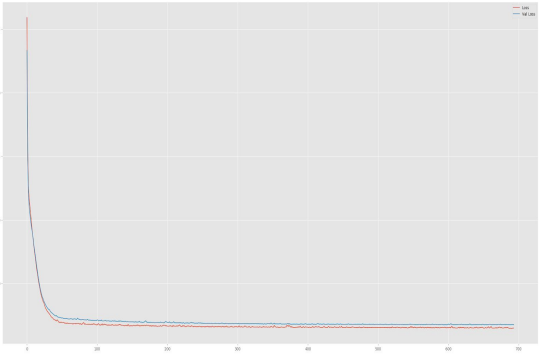
3.1 Single-Step

	Dati Orari	Dati Giornalieri
Activation Function	relu	relu
Optimizer	Adam	Adam
# Layers	1	1
Neurons per Layer	1.024	2.048
Features	Ore al Tramonto Ferie Sabato Domenica Orario Lavorativo PUNTORUGIADA C Moving Average	Tramonto Feste Weekend PUNTORUGIADA C Moving Average

One Day Forecast *One Hour Forecast*

Andamento Training





Codice Sorgente

```
In [196]: input_shape = (X.shape[1],)
          model = tf.keras.Sequential()
          model.add(tf.keras.layers.Dense(units=2048, input_shape=input_shape,activation=tf.keras.activations.relu,
                                          bias_initializer='zeros', kernel_initializer='random_normal'))
          model.add(tf.keras.layers.Dense(units=1, activation=tf.keras.activations.linear, bias_initializer='zeros',
                                          kernel_initializer='random_normal'))

          loss = tf.keras.losses.mean_absolute_error
          model.compile(loss=loss, optimizer='adam', metrics=['mae','mape'])
          callbacks = [
              tf.keras.callbacks.EarlyStopping(patience=100, verbose=1),
              tf.keras.callbacks.ModelCheckpoint('model.h5', verbose=1, save_best_only=True, save_weights_only=True)
          ]

          val_split = 0.31
          history = model.fit(X, Y, epochs=2000, validation_split=val_split, callbacks=callbacks, verbose=1)
```

```
In [278]: input_shape = (X.shape[1],)
          model = tf.keras.Sequential()
          model.add(tf.keras.layers.Dense(units=1024, input_shape=input_shape,activation=tf.keras.activations.relu,
                                          bias_initializer='zeros', kernel_initializer='random_normal'))
          model.add(tf.keras.layers.Dense(units=1, activation=tf.keras.activations.linear, bias_initializer='zeros',
                                          kernel_initializer='random_normal'))

          loss = tf.keras.losses.mean_absolute_percentage_error
          model.compile(loss=loss, optimizer='adam', metrics=['mape', 'mae'])
          callbacks = [
              tf.keras.callbacks.EarlyStopping(patience=100, verbose=1),
              tf.keras.callbacks.ModelCheckpoint('model.h5', verbose=1, save_best_only=True, save_weights_only=True)
          ]

          val_split = 0.3
          history = model.fit(X, Y, epochs=2000, validation_split=val_split, callbacks=callbacks, verbose=1)
```

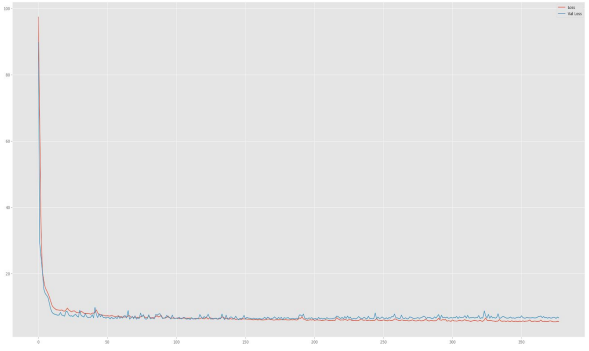
3.2. Multi-Step

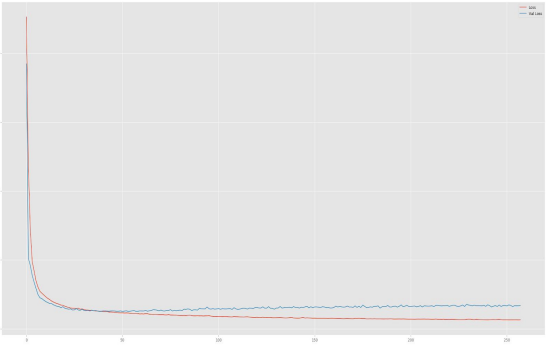
	Dati Orari	Dati Giornalieri
Activation Function	relu	relu
Optimizer	Adam	Adam
# Layers	2	2
Neurons per Layer	1.024/512	512/256
Features	Ore al Tramonto Ferie Sabato Domenica Orario Lavorativo PUNTORUGIADA C Moving Average	Consumo Tramonto Feste Weekend PUNTORUGIADA C Moving Average
Forecast Horizon	24h	7gg

Trattandosi di una previsione multi-step la moving average si riferisce solo ai giorni precedenti al giorno previsto.

One Week Forecast *24h Forecast*

Andamento Training





Codice Sorgente

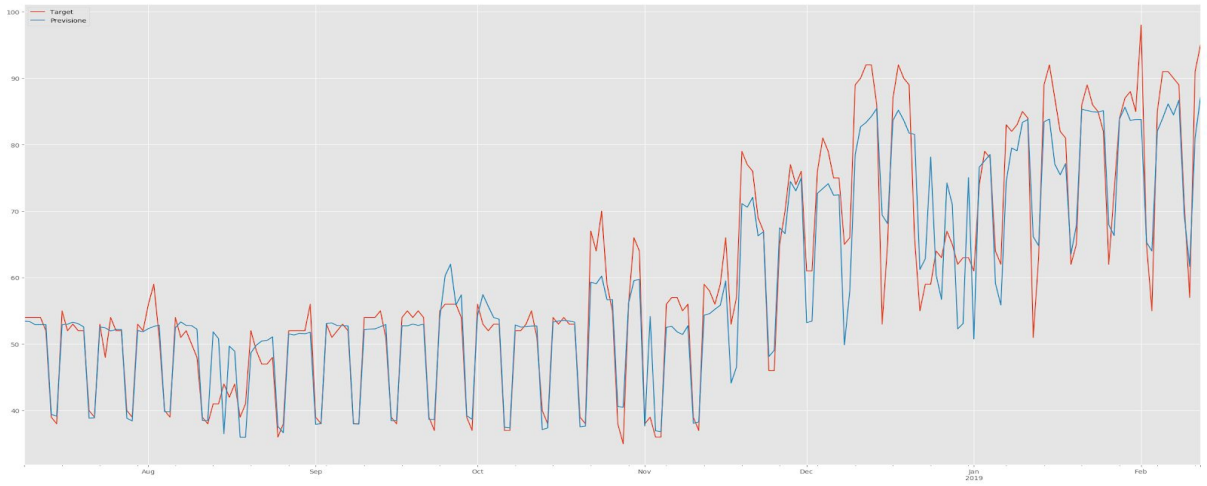
```
In [298]: input_shape = (X.shape[1], X.shape[2])
          kernel_initializer = 'random_normal'
          hidden_initialization = tf.keras.activations.relu
          model = tf.keras.Sequential()
          model.add(tf.keras.layers.Dense(units=512, input_shape=input_shape, activation=hidden_initialization,
                                          bias_initializer='zeros', kernel_initializer='random_normal'))
          model.add(tf.keras.layers.Dense(units=256, input_shape=input_shape, activation=hidden_initialization,
                                          bias_initializer='zeros', kernel_initializer='random_normal'))
          model.add(tf.keras.layers.Flatten())
          model.add(tf.keras.layers.Dense(units=finestra, activation=tf.keras.activations.linear, bias_initializer='zeros',
                                          kernel_initializer='random_normal'))

          loss = tf.keras.losses.mean_absolute_error
          model.compile(loss=loss, optimizer='adam', metrics=['mae', 'mape'])
          callbacks = [
              tf.keras.callbacks.EarlyStopping(patience=200, verbose=1),
              tf.keras.callbacks.ModelCheckpoint('mlp_week.h5', verbose=1, save_best_only=True, save_weights_only=True)
          ]

          val_split = 0.31
          history = model.fit(X, Y, epochs=2000, validation_split=val_split, callbacks=callbacks, verbose=1)
```

```
In [233]: input_shape = (X.shape[1], X.shape[2])
          kernel_initializer = 'random_normal'
          hidden_initialization = tf.keras.activations.relu
          model = tf.keras.Sequential()
          model.add(tf.keras.layers.Dense(units=1024, input_shape=input_shape, activation=hidden_initialization,
                                          bias_initializer='zeros', kernel_initializer='random_normal'))
          model.add(tf.keras.layers.Flatten())
          model.add(tf.keras.layers.Dense(units=finestra, activation=tf.keras.activations.linear,
                                          bias_initializer='zeros', kernel_initializer='random_normal'))

          loss = tf.keras.losses.mean_absolute_error
          model.compile(loss=loss, optimizer='adam', metrics=['mae', 'mape'])
          callbacks = [
              tf.keras.callbacks.EarlyStopping(patience=200, verbose=1),
              tf.keras.callbacks.ModelCheckpoint('mlp_24_hours.h5', verbose=1, save_best_only=True, save_weights_only=True)
          ]

          val_split = 0.31
          history = model.fit(X, Y, epochs=2000, validation_split=val_split, callbacks=callbacks, verbose=1)
```

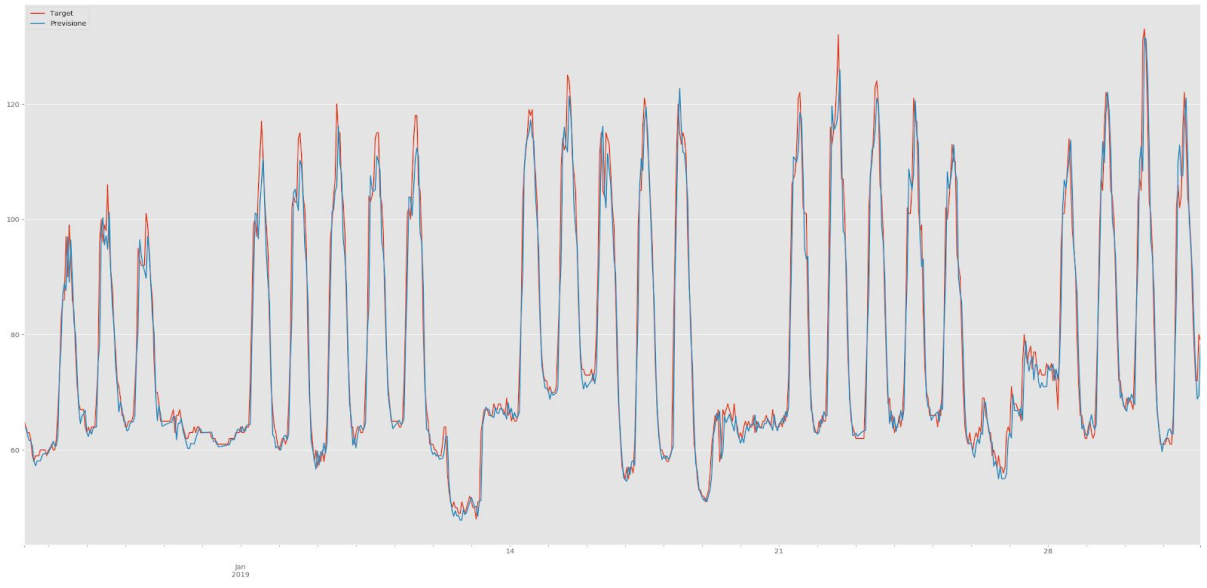
4. Risultati

	MAPE	MAE
1d	5.87	3.55
1h	3.47	3.02
7dd	5.96	3.58
24h	5.02	4.09

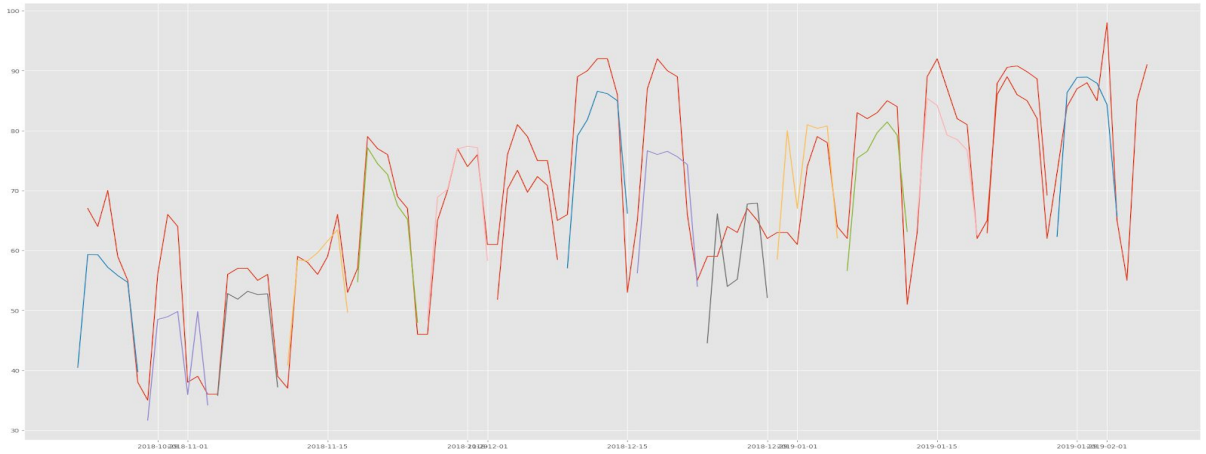
Previsione Giornaliera



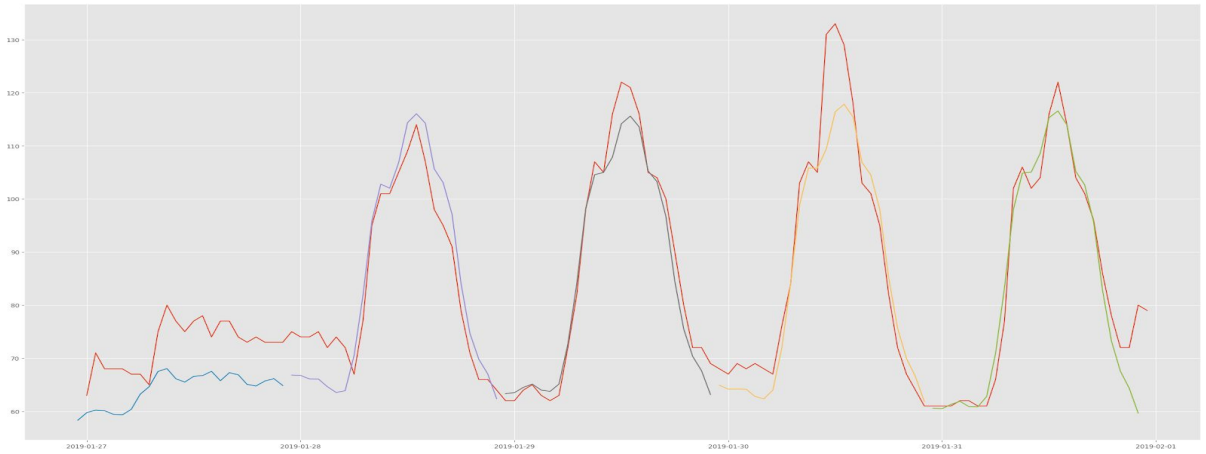
Previsione Oraria



Previsione 7gg



Previsione 24h



5. Conclusioni

I risultati del progetto hanno permesso alcune conclusioni interessanti. Riguardo ai dati riguardo le condizioni meteorologiche è risultato, come del resto ci si poteva aspettare, che la temperatura esterna incida in modo pesante sui risultati. In particolare il cosiddetto ‘Punto Rugiada’ sembra il migliore parametro da tenere in considerazione per prevedere i consumi, probabilmente perché tiene contemporaneamente in considerazione temperatura e umidità. Sorprendente è stata invece lo scarso contributo delle altre variabili meteo, infatti parametri come visibilità, intensità del vento e fenomeni meteorologici hanno avuto un impatto scarso o nullo sulla precisione del modello.
Per quanto riguarda l’efficacia del modello stesso, la conclusione che mi sento di trarre è che il numero relativamente basso di dati, combinato alla presenza di alcuni outlier nei dati

(ad esempio la differenza nell’andamento dei dati tra natale 2017 e natale 2018) non abbia permesso di sfruttare al meglio la potenza espressiva delle reti neurali implementate, i cui risultati non si discostano significativamente da quelli ottenuti con metodi autoregressivi classici.