Università degli Studi di Milano-Bicocca

Streaming Data Management and Time Series Analysis
Final Project

Forecast Time Series

Autore:
Massimiliano Perletti - 847548 - m.perletti2@campus.unimib.it

28 gennaio 2020

Indice

Sommario

In questo lavoro vengono proposti una serie di modelli tra quelli lineari (ARIMA e UCM) e alcuni di machine learning (GRU e LSTM) volti ad affrontare un task di predizione di serie temporali riguardanti l'andamento dei prezzi del mercato elettrico italiano. I modelli verranno valutati e confrontati sulla base della metrica MAPE, e tramite una valutazione grafica. Sarà possibile quindi vedere come si comportano i diversi modelli con la serie storica data, portando ad alcune interessanti osservazioni.

1 Introduzione

L'obiettivo di questo progetto è quello di modellare dei sistemi predittivi su delle serie temporali che riguardano l'andamento dei prezzi del mercato elettrico italiano. Lo scopo è quello di testare diverse metodologie, tra cui un *modello autoregressivo integrato a media mobile* (ARIMA), un *modello a componenti non osservabili* (UCM) e una rete neurale ricorsiva (RNN). È richiesta una previsione nel periodo dal **1-Gen-2019** al **30-Nov-2019**.
La serie (raffigurata in Fig. 1) rappresenta un'aggregazione giornaliera dei prezzi del mercato energetico (criterio di aggregazione non noto). Il periodo di riferimento di questi dati è dal **1-Gen-2010** al **31-Dic-2018**.

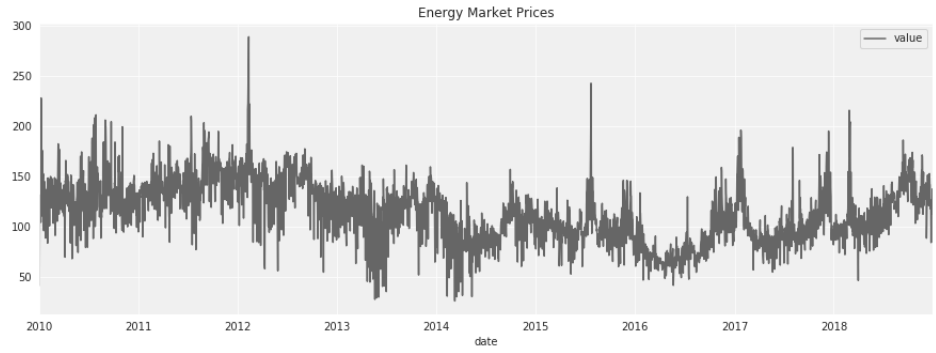

Figura 1: Time Series

È possibile notare alcuni picchi sulla serie, che non verranno però trattati, in quanto portatori di probabili eventi anomali sulla serie, e quindi da considerare. La condizione di stazionarietà è verificata tramite *Dickey-Fuller test*, che rifiuta l'ipotesi nulla. Sarebbe possibile effettuare alcune trasformazioni (come *log* oppure *box-cox*), ma dopo alcuni test sulla performance che non hanno portato a particolari miglioramenti, si è optato per tenere la serie nella sua forma originale.
Per la divisione del **training set** e **validation set** (vedi Fig. 2) è stato deciso per tutti i modelli di tenere otto anni per l'addestramento e un solo anno per la validazione sulle previsioni necessarie per lo sviluppo dei modelli; si è cosi ottenuta una proporzione rispettivamente del **88.9% - 11.1%**.

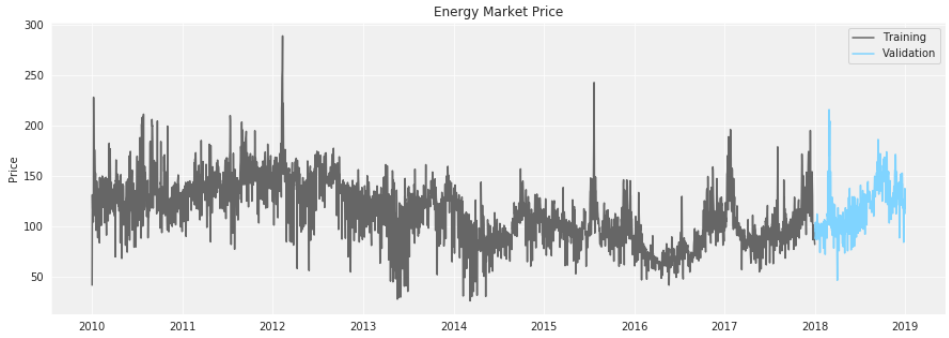

Figura 2: Divisione Training set e Validation set

Per la valutazione delle metodologie applicate, e per poter applicare in seguito un loro confronto, è stato scelta la metrica **MAPE**, in quanto è una metrica facilmente interpretabile in termini di errore dal punto di visto umano (quanto sovra/sotto predice il modello rispetto al valore reale?). Inoltre, verrà comunque esaminata la predizione anche dal punto di vista grafico, in quanto può capitare che a valori di MAPE bassi corrispondano linee rette, o che comunque non seguano adeguatamente le previsioni reali; si cercherà dunque un giusto trade-off tra le due analisi.

2 Arima

Si esegue inizialmente un'analisi tramite *autocorrelation function* (ACF) e *partial autocorrelation function* (PACF) (Fig. 8 Appendice A.1) dove si vede chiaramente l'esistenza di tutte e tre le componenti stagionali, con stagionalità 7 (discesa lineare ogni sette giorni in ACF e discesa geometrica ogni sette giorni in PACF). Si studia quindi un modello $\text{AR}(1)_7\text{I}(1)_7\text{MA}(1)_7$, andando ad analizzare le componenti non stagionali probabilmente presenti.
Per determinare dunque i parametri p,q dell'ARIMA (AR(p)I(1)MA(q)) si esegue tramite **Grid Search** la ricerca del modello che massimizzi la *log-likelihood*, con parametri $p = 0, \dots 6$ e $q = 0, \dots 6$. Si trova così il modello migliore, ossia un $\text{ARIMA}(6, 1, 6)(1, 1, 1)_7$, dove l'ACF e la PACF (Fig. 9 Appendice A.1) risultano adeguatamente modellati.
Vengono infine provati dei modelli con dei regressori esterni, nello specifico tre diverse matrici di regressori:

- FESTE, dodici feste del calendario italiano
- FREQ, sin e cos del seguente array $[2\pi * seq(dayOftheMonth)/365.25]$
- integrazione FESTE e FREQ

Vengono valutati rispetto al valore del MAPE in Validation. Risulta così che il modello migliore sia quello con i regressori relativi alle feste, ottenendo un valore del **MAPE** (validation) pari a **13.83%**. Nella Fig. 3 è possibile vedere le predizioni sul Training e Validation set del modello.

Figura 3: Predizioni modello ARIMA(6, 1, 6)(1, 1, 1)$_7$

3 Ucm

Per lo sviluppo di questo modello si è deciso di iniziare l'analisi usando come trend il *local level*, aggiungendo una componente ciclica e una stagionalità stocastica settimanale (dummy). Vedendo che tutte le componenti erano significative, ma il MAPE risultava piuttosto alto (circa 19.44%), si è deciso di trovare la componente trend migliore sempre tramite approccio **Grid Search**, vedendo quale modello ottenesse MAPE più bassi; è stato così selezionato il modello che prevedeva un livello con *Random Walk Integrato*.
In seguito, sono stati testati diversi modelli con differenti componenti stagionali:

- ciclo, stagionalità 7 dummy e stagionalità 30 (mensile) trigonometrica (12 armoniche)
- ciclo, stagionalità 7 dummy, stagionalità 365 (annua) trigonometrica (24 armoniche)

- ciclo, stagionalità 7 dummy, stagionalità 30 e 365 (mensile e annua) trigonometrica (12 e 24 armoniche)
- senza ciclo, stagionalità 7 dummy, stagionalità 30 e 365 (mensile e annua) trigonometrica (12 e 24 armoniche)

In questo caso il modello migliore è risultato essere quello con la componente ciclo, e la stagionalità settimanale dummy e annua trigonometrica. Infine, come per il modello ARIMA, sono stati provati i modelli con tutti i regressori precedentemente descritti, senza però ottenere nessun miglioramento. È stato così ottenuto il modello UCM finale, che ha ottenuto un valore del **MAPE** (validation) pari a **14.95%**. In Fig. 4 è possibile vedere graficamente le predizioni effettuate dal modello.

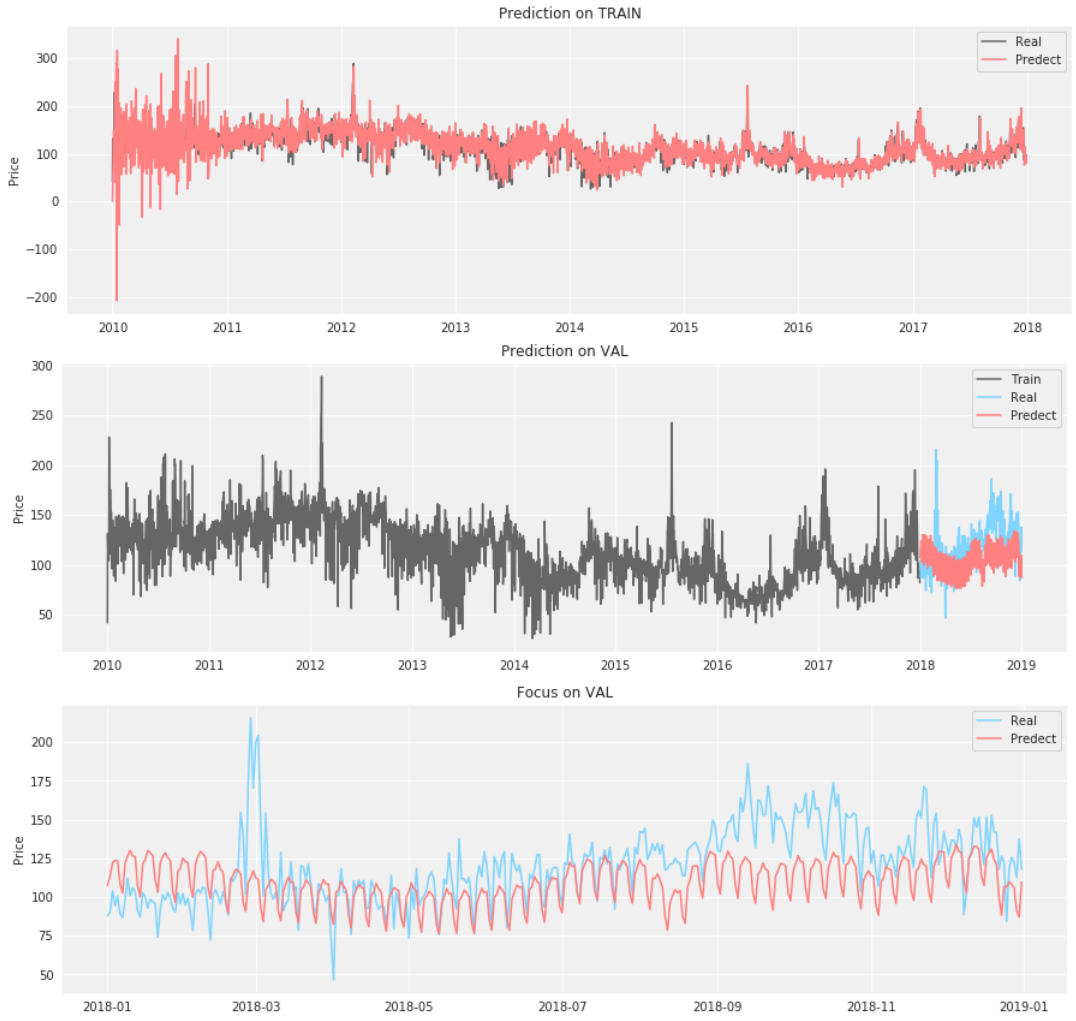

Figura 4: Predizioni modello UCM

4 Machine Learning

Per valutare le performance di un modello di machine learning, e nello specifico mediante l'utilizzo di una RNN, è stato deciso di testare due architetture: una **GRU** e una **LSTM**. Per il pre-processing dei dati è stato utilizzato il *MinMaxScaler*, per poter scalare i dati un un range [0.001, 1]. Non viene scelto zero come limite inferiore in quanto questo causerebbe problemi nella metrica del MAPE sulla valutazione del training durante l'addestramento; infatti se un valore reale della serie è approssimato a zero, il calcolo del MAPE ($abs\|Ypred - Yreale\|/Yreale$) esploderebbe. Sono stati ovviamente scalati rispetto al training set.
Per la creazione della time series da dare al modello per l'addestramento, si è utilizzata una finestra che guarda indietro (chiamata *look_back*) di 365 giorni. Si sono così create dunque tutte le finestre possibili "dividendo" il training in finestre. Come target da predire che viene passato al modello, si usa il valore puntuale successivo alla finestra (look_back + 1). Dopodiché si è proceduto con la realizzazione dell'architettura, cosi composta:

- layer **GRU** oppure **LSTM**, con 256 neuroni e attivazione $LeakyReLU$
- **Dropout**(rate = 0.33)
- **BatchNormalization**
- layer **Dense**, con 256 neuroni e attivazione $ReLU$
- **Dropout**(rate = 0.33)
- output **Dense**, 1 neurone e attivazione *sigmoid*

Per la compilazione del modello si è usata come loss **mse**, ottimizzatore **Adam** e come metrica **mape**. Sono stati usati layer *CuDNNGRU* e *CuDNNLSTM*, del pacchetto keras, in quanto ottimizzati per la compilazione tramite GPU; questo ha permesso una velocità di addestramento di circa 10 volte più veloce rispetto ai layer canonici. È stata impostata una batch_size pari a 512, e un numero di epoche pari a 500. Considerata l'assenza, per come costruita la serie, del validation nell'addestramento, è stata usata come callback il *ModelCheckpoint* che monitorava il modello salvando i pesi relativi al valore più basso possibile del MAPE in training.
Sono state così confrontate le due architetture, che ha portato a scegliere il modello GRU in quanto riesce ad ottenere valori migliori con un numero di parametri e complessità minore rispetto a LSTM. Si è ottenuto così il modello finale con un valore del **MAPE** (validation) pari a **13.32%**. In Fig. 5 è possibile vedere graficamente le predizioni effettuate dal modello.

Figura 5: Predizioni modello GRU

5 Conclusioni

Come è possibile vedere nella Tab. 1, le performance sono molto simili tra i diversi modelli, con quello GRU che riesce ad ottenere valori migliori sia in training che validation. Può essere un risultato comprensibile, basti pensare che le reti sono in grado di cogliere in maniera autonoma le varie parti della stagionalità e del trend, andando a minimizzare l'errore su l'intera serie, mentre nei modelli lineari come ARIMA e UCM, le componenti hanno bisogno di una modellazione controllata e quindi più sensibile ad errori umani.
I modelli finali ottenuti vengono, infine, riaddestrati utilizzando tutta la serie a disposizione, per effettuare le previsioni oggetto del progetto. Possiamo vedere nella Fig. 6 e Fig. 7 i vari andamenti delle previsioni sulla serie.

Modello	ARIMA	UCM	GRU
Training	9.78%	13.03%	**8.65%**
Validation	13.83%	14.95%	**13.32%**

Tabella 1: Risultati MAPE dei Modelli

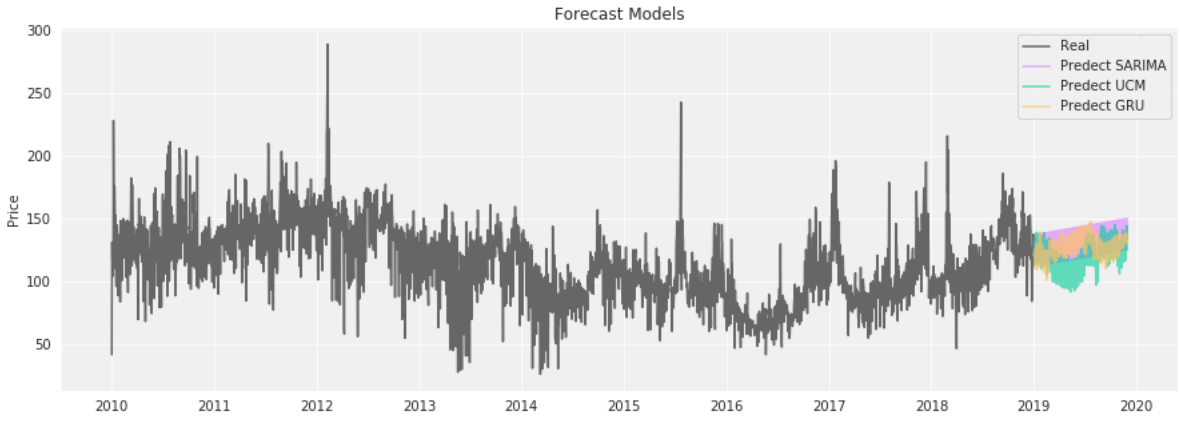

Figura 6: Predizioni sulla serie

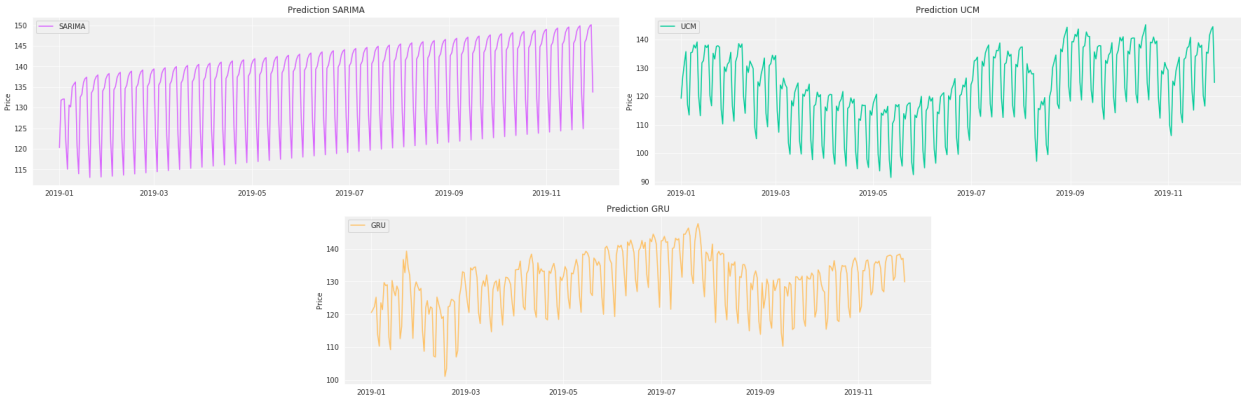

Figura 7: Focus predizioni sulla serie

Si nota come tendenzialmente il modello GRU sembrerebbe seguire più l'andamento globale della serie (appunto perché l'ha memorizzata per minimizzare l'errore), mentre il modello UCM segue di più l'andamento degli ultimi anni. Infine, il modello ARIMA che sembra aver colto bene la stagionalità settimanale, ma meno quella annuale.

A Appendix: Grafici

A.1 ARIMA

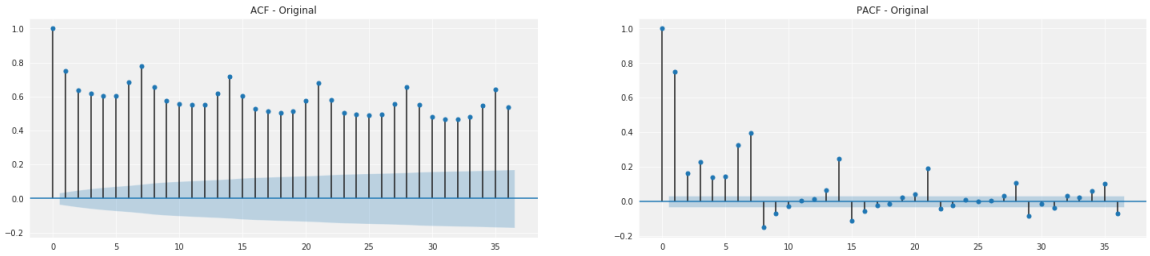

Figura 8: ACF e PACF della time series

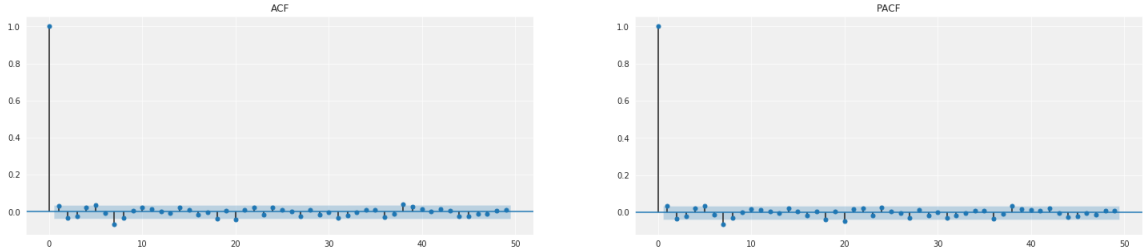

Figura 9: ACF e PACF ARIMA$(6, 1, 6)(1, 1, 1)_7$

```
<<<<<<<<<<  Senza Regressori  <<<<<<<<<<

MAPE Train: 9.785313999565744
MAPE Validation: 13.857234338988814

<<<<<<<<<<  REG - Feste  <<<<<<<<<<

MAPE Train: 9.782285943979293
MAPE Validation: 13.828366756108274

<<<<<<<<<<  REG - Freq  <<<<<<<<<<

MAPE Train: 9.797123565663357
MAPE Validation: 13.877569482538457

<<<<<<<<<<  REG - Feste + Freq  <<<<<<<<<<

MAPE Train: 9.261652205970952
MAPE Validation: 14.41084720087986
```

Figura 10: MAPE sul modello ARIMA finale con diversi regressori esterni

A.2 UCM

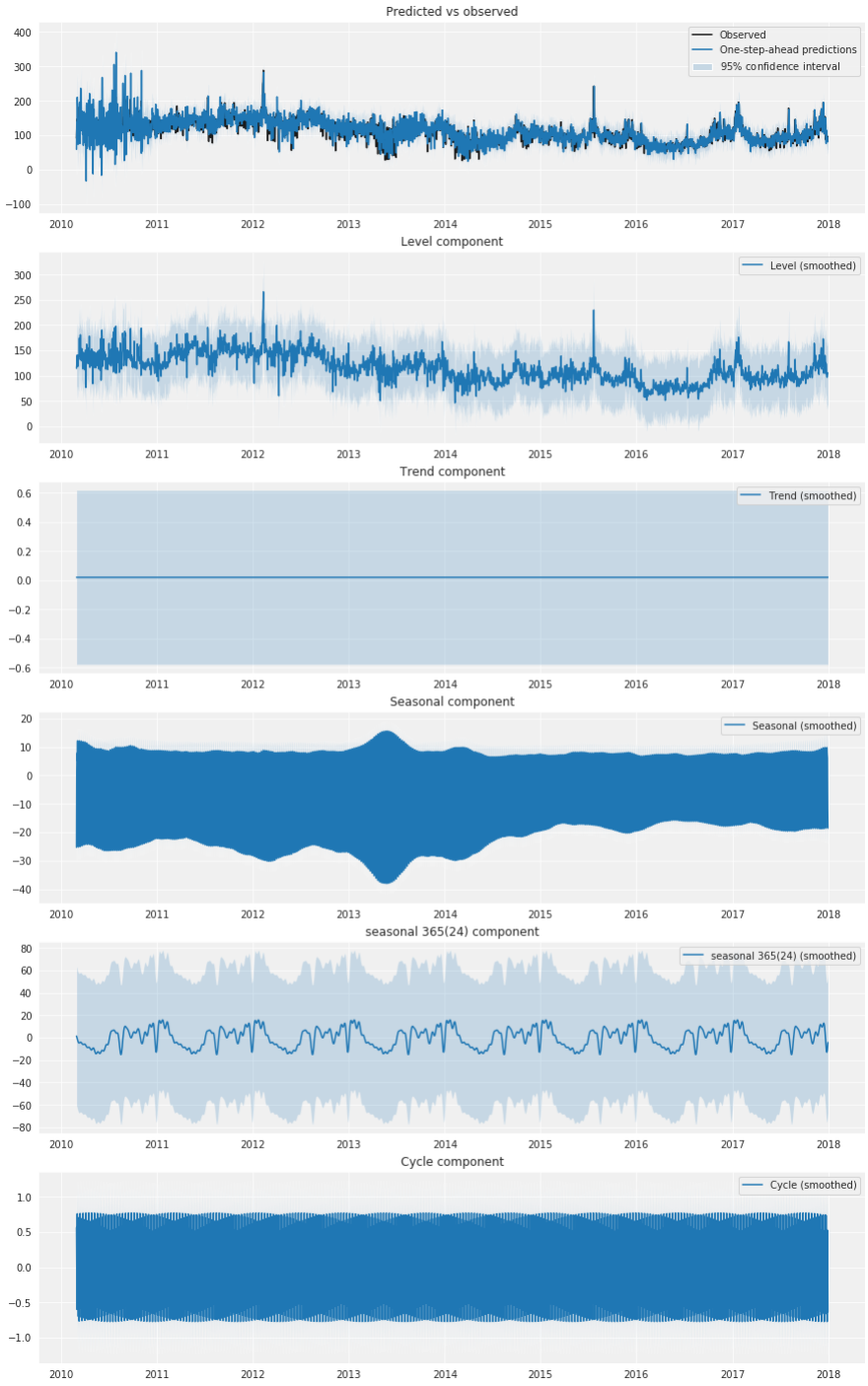

Figura 11: Alcune diagnostiche del modello finale UCM

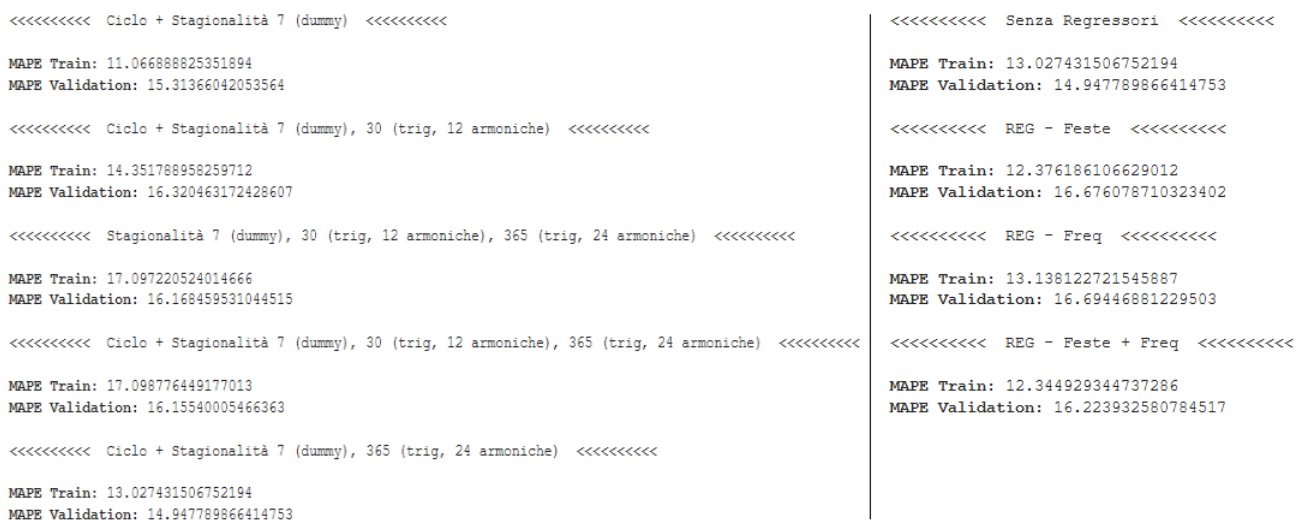

Figura 12: MAPE sul modello UCM finale con diverse stagionalità e regressori esterni

A.3 RNN

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cu_dnngru_1 (CuDNNGRU)       (None, 256)               478464
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 545,537
Trainable params: 545,025
Non-trainable params: 512
_________________________________________________________________
<keras.engine.sequential.Sequential at 0x7f8ad51e3198>
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cu_dnnlstm_1 (CuDNNLSTM)     (None, 256)               637952
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 705,025
Trainable params: 704,513
Non-trainable params: 512
_________________________________________________________________
<keras.engine.sequential.Sequential at 0x7f8ad51cf550>
```

Figura 13: Architettura modelli GRU (sinistra) e LSTM (destra)

```
MAPE Train: 8.650046769751562
MAPE Validation: 13.321635416767197
```

```
MAPE Train: 9.774944204057011
MAPE Validation: 13.499660970939475
```

Figura 14: MAPE sul modello GRU (alto) e LSTM (basso)

B Appendix: Codice

È possibile visionare il codice completo per lo sviluppo di tutti i modelli, tramite la repository disponibile su GitHub, al seguente indirizzo: `https://github.com/maxi93/SDMTSA`