PROGETTO ESAME DI STREAMING DATA MANAGEMENT AND TIME SERIES ANALYSIS

Davide Mancino, d.mancino1@campus.unimib.it, Matricola: 847700

Prof. Matteo Pelagatti e Prof. Antonio Candelieri
AA 2020/21

Sinossi. Questo progetto ha come obiettivo quello di sviluppare un sistema predittivo per una serie storica, nel quale sono stati testati e validati i modelli ARIMA, TBATS, LSTM, PROPHET e UCM. Si è cercato di definire e sviluppare i modelli osservando gli andamenti stagionali, annuali, mensili, settimanali e giornalieri della serie storica data. Infine, si è osservato come il lockdown dovuto alla pandemia da Covid19 abbia influenzato l'andamento della stessa.

Aspetti metodologici. I passaggi dell'approccio metodologico utilizzato sono stati:

1. studio dei dati a disposizione;
2. preprocessing e data cleaning: gestione dati missing, anomalie e sui valori uguali a 0;
3. visualizzazioni delle serie storiche per avere un quadro completo dell'andamento delle stesse;
4. visualizzazioni più specifiche riguardanti l'andamento dei dati in relazione ad archi temporali differenti, ad esempio andamento settimanale e andamento giornaliero;
5. sviluppo dei modelli;
6. ottimizzazione dei parametri dei modelli;
7. validazione dei modelli;
8. visualizzazione delle previsioni sui dati del validation set dei modelli più performanti;
9. commento sui dati ottenuti;
10. estrazione di informazioni rilevanti presenti nella serie storica.

Analisi/Processo di trattamento dei dati. I dati a disposizione provengono da un unico file (formato .csv) messo a disposizione dai docenti di questo corso per l'elaborato finale. All'interno del dataset troviamo dati che rappresentano delle misurazioni, con cadenza oraria, dal 2018-09-01 fino al 2020-08-31.
Le colonne a disposizione sono:

- DATA: giorno di rilevazione (formato: aaaa-mm-gg);
- ORA: valore progressivo che indica l'orario di rilevazione. Le rilevazioni avvengono ogni ora. Inoltre, le ore vanno dall'1 alle 24 (tipologia dato: numero intero);
- VALORE: questa colonna rappresenta la nostra variabile di interesse (tipologia dato: numero decimale);

Il dataset presenta 17.518 osservazioni.
Le rilevazioni ogni ora ci permettono di avere una buona rappresentazione dell'andamento giornaliero. Questo rappresenta anche un punto di forza, poiché, fornisce la possibilità di analizzare e prendere in considerazione diversi archi temporali, come le informazioni aggregate per giorno, settimana, mese e anno.
L'import dei file e l'intero progetto è stato svolto in Python. Durante questa prima fase, la libreria principalmente utilizzata è stata Pandas. Successivamente, le colonne di DATA e ORA sono state trasformate

in un'unica colonna denominata *DATA_ORA* la quale contiene le informazioni di data e ora delle rilevazioni in formato datetime YYYY-MM-DD HH:MM:SS.

es. DATA= *20180101*, ORA= *1* → DATA_ORA= *2018-01-01 00:00:00*

La colonna ORA che andava da un range 1 a 24 è stata trasformata in un range di dati che va da 0 a 23. In seguito, si è passati ad analizzare eventuali problemi all'interno dei dati, ad esempio:

- rilevazioni duplicate per DATA_ORA;
- rilevazioni mancanti per alcune date;
- valori rilevati uguali a 0.

Per risolvere la prima anomalia, si è proceduto identificando le rilevazioni duplicate per medesima data e ora e si è potuto osservare come nessuna fosse duplicata.
Per gestire il secondo problema, è stato recuperato l'orario e le date di tutte le rilevazioni non presenti all'interno dei dati importati. Le seguenti osservazioni distoniche sono dovute soprattutto al cambio di orario, presente nei mesi di marzo con il passaggio da ora solare a ora legale. In questo caso, le rilevazioni da bonificare erano solamente 2 ('2019-03-31 03:00:00', '2020-03-29 03:00:00'), si è deciso di inserire come valore mancante quello rilevato durante l'ora precedente. Essendo rilevazioni notturne e non essendoci grosse differenze, è stata scelta la precedente soluzione in modo da non alterare l'andamento della serie storica (Fig. 1 e 2).
Inoltre, una intera giornata non era presente all'interno del dataset iniziale ovvero il 2020-07-31. In questo caso, essendo una intera giornata, si è deciso di inserire i valori medi corrispondenti alle ore di rilevazione tra quello della settimana precedente e quello della settimana successiva (Fig. 3 e 4).
Così facendo si è cercato di inserire dei valori consoni sfruttando l'andamento simile che la serie storica assume ogni settimana.
Il terzo ed ultimo problema è stato quello di risolvere le rilevazioni uguali a 0, per prevenire eventuali errori successivi dovuti a eventuali trasformazioni logaritmiche. Per questo tema ci si è resi conto che nessuna rilevazione assume come valore lo 0.

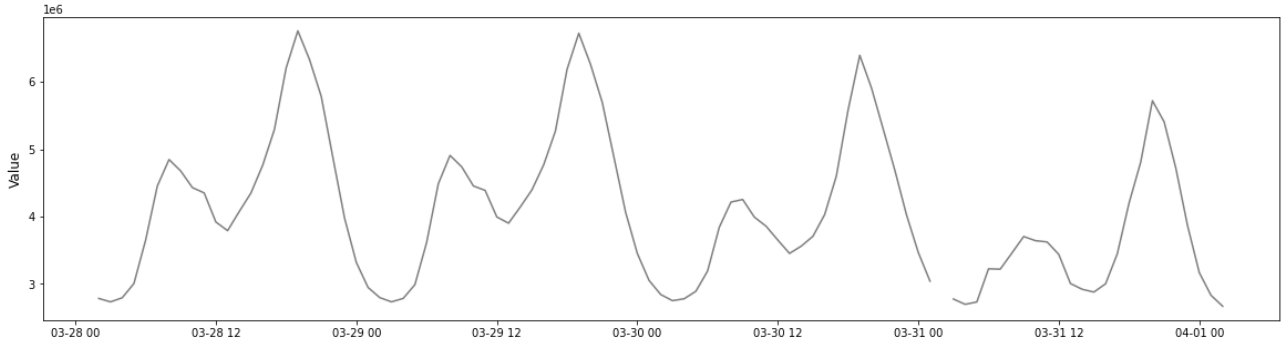

Fig. 1.: all'interno del dataset iniziale per il 2019-03-31 alle 03:00 non è presente nessuna rilevazione.

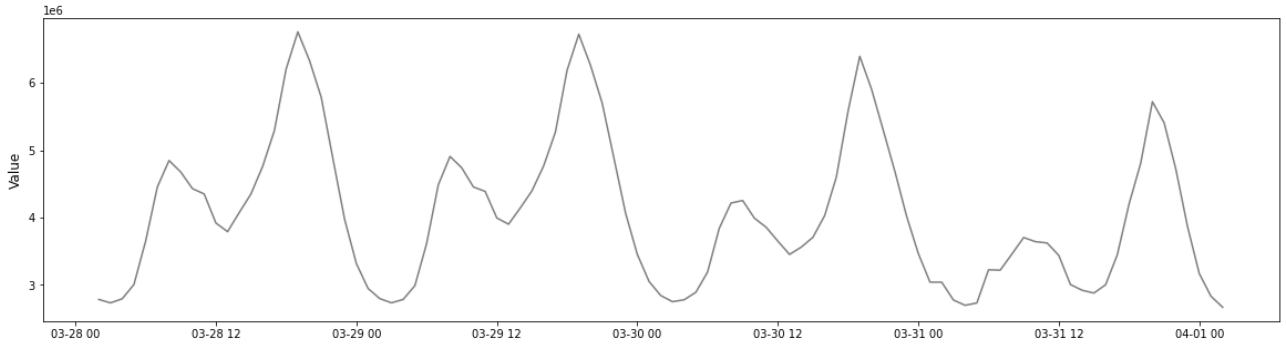

Fig. 2.: rilevazione del 2019-03-31 alle 03:00 bonificata.

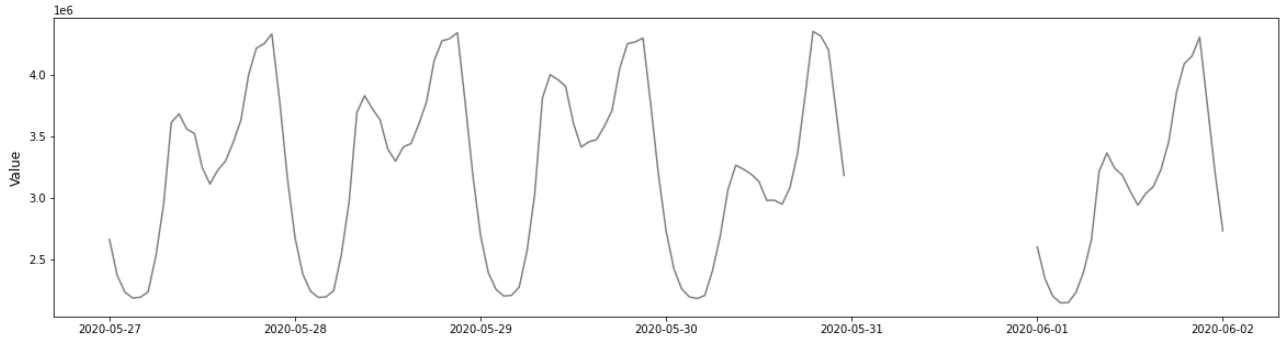

Fig. 3.: dati originari giornata del 2020-05-31.

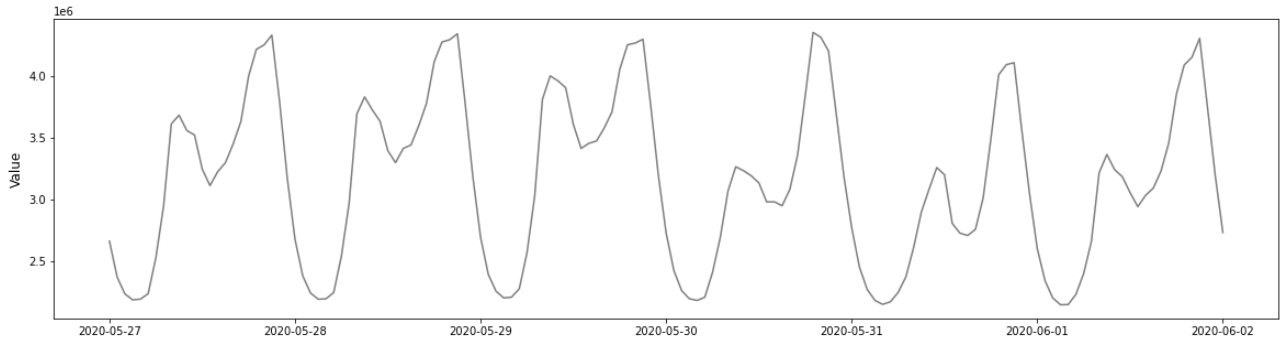

Fig. 4.: dati post bonifica della giornata del 2020-05-31.

Completate queste operazioni di data cleaning e preprocessing, è stato salvato un file .csv contenente il dataset completo.
Fatto ciò, ci si è focalizzati sulle visualizzazioni iniziali per estrarre possibili archi temporali critici, per controllarne gli andamenti e per provare a estrarre informazioni rilevanti.
La figura 5. rappresenta la serie storica completa a partire dal 2018-09-01 fino al 2020-08-31. Inoltre, i box plot della figura 6. ci forniscono delle informazioni sull'andamento annuale e mensile della serie storica.
Osservando le figure 6 e 7, possiamo notare un trend decrescente, da sottolineare, però, che i dati riferiti al 2020 sono probabilmente influenzati dal lockdown di marzo 2020 dovuto alla pandemia da Covid19. Dalle due figure precedenti emergono dei picchi presenti nei mesi di gennaio, luglio e dicembre.
I box plot mensili ci danno già una indicazione di un possibile andamento annuale confermato anche dai grafici presenti nella figura 7.

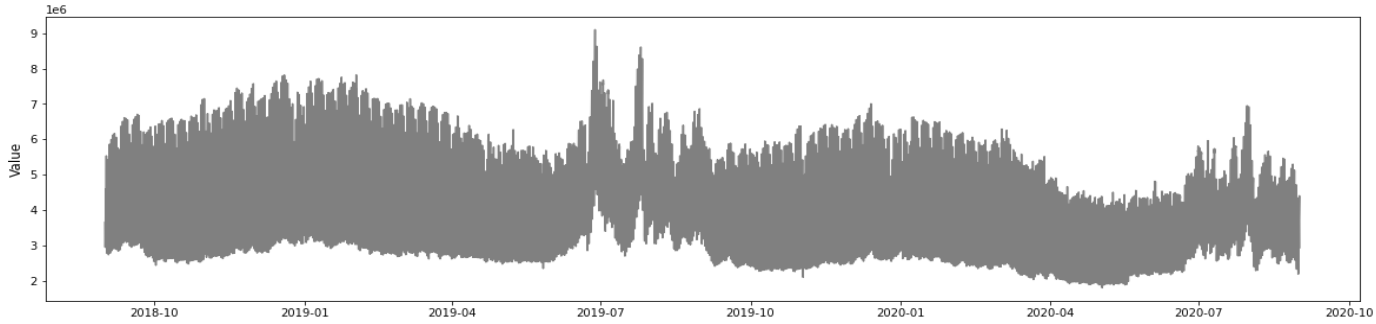

Fig. 5.: Serie storica completa.

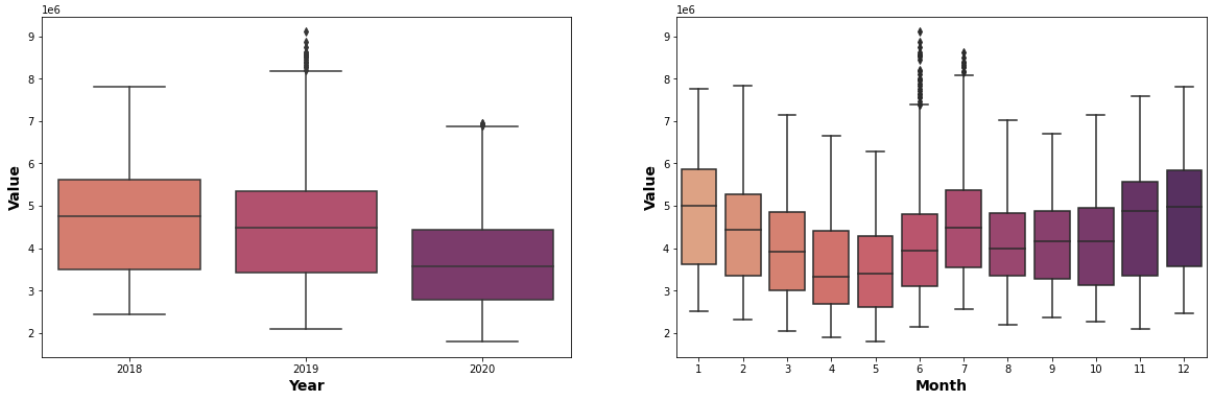

Fig. 6.: Serie storica edificio U6, Box plot annuali e mensili.

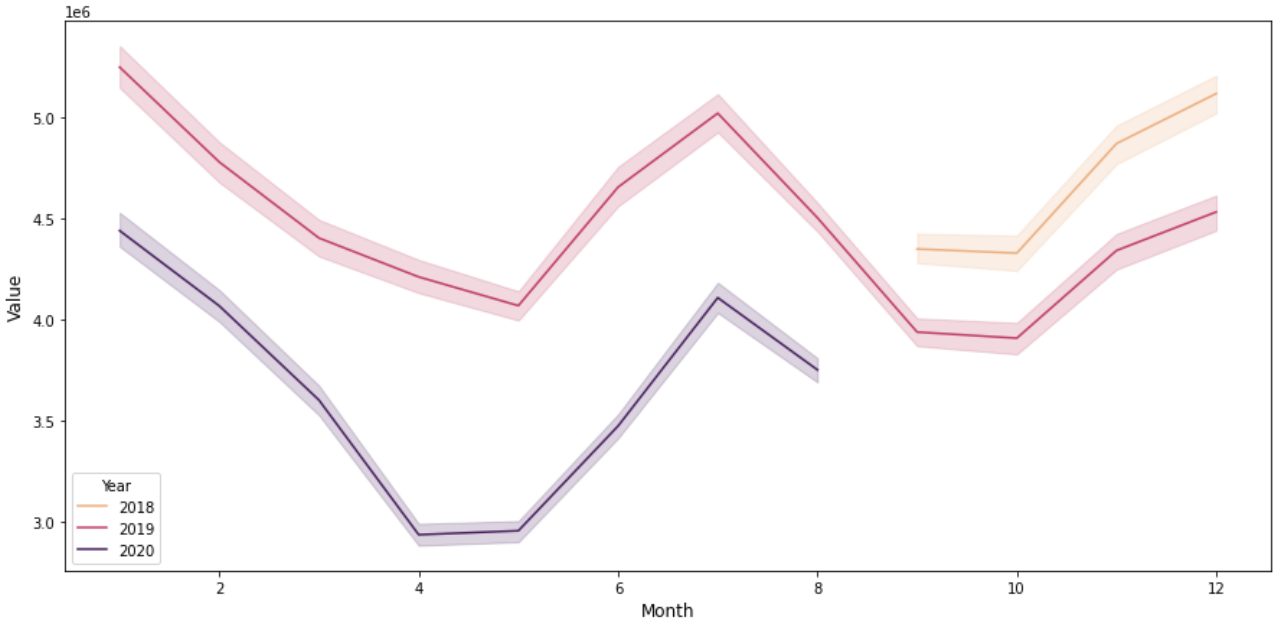

Fig 7.: stagionalità annuale

Oltre all'analisi sulla stagionalità annuale sono state fatte delle analisi su:

1. andamento dei giorni della settimana (Figura 8.);
2. andamento dei consumi per specifica ora al variare dei giorni della settimana (Figura 9.);

3. andamento dei consumi per specifica ora al variare dei mesi (Figura 10.).

Da questi grafici si possono estrarre numerose informazioni. Ad esempio, osservando gli andamenti presenti all'interno della Figura 8. (che rappresentano l'andamento settimanale), si può verificare come i valori assunti dalla serie storica siano molto simili dal lunedì al venerdì, mentre rilevano una repentina decrescita durante i weekend. Un'altra conferma a favore di questa tesi si può ritrovare all'interno della figura 9, nella quale si osserva l'andamento orario per giorno della settimana. La stessa, infatti, oltre a dimostrare come i valori registrati tra sabato e domenica siano inferiori rispetto a quelli presenti nel resto della settimana, evidenzia come nelle prime ore del mattino (dalle 03:00 alle 04:00) vengano registrati i valori più bassi. Inoltre, durante la giornata si registrano due picchi, alle 08:00 e alle 19:00. Per la giornata di domenica i picchi vengono registrati in leggero ritardo, ovvero, alle 11:00 e alle 20:00.

La figura 10 presenta gli andamenti orari registrati nei vari mesi. Il grafico indica come i mesi di gennaio, febbraio, luglio e dicembre siano quelli che presentano una crescita repentina con picchi massimi maggiori verso l'ora di punta.

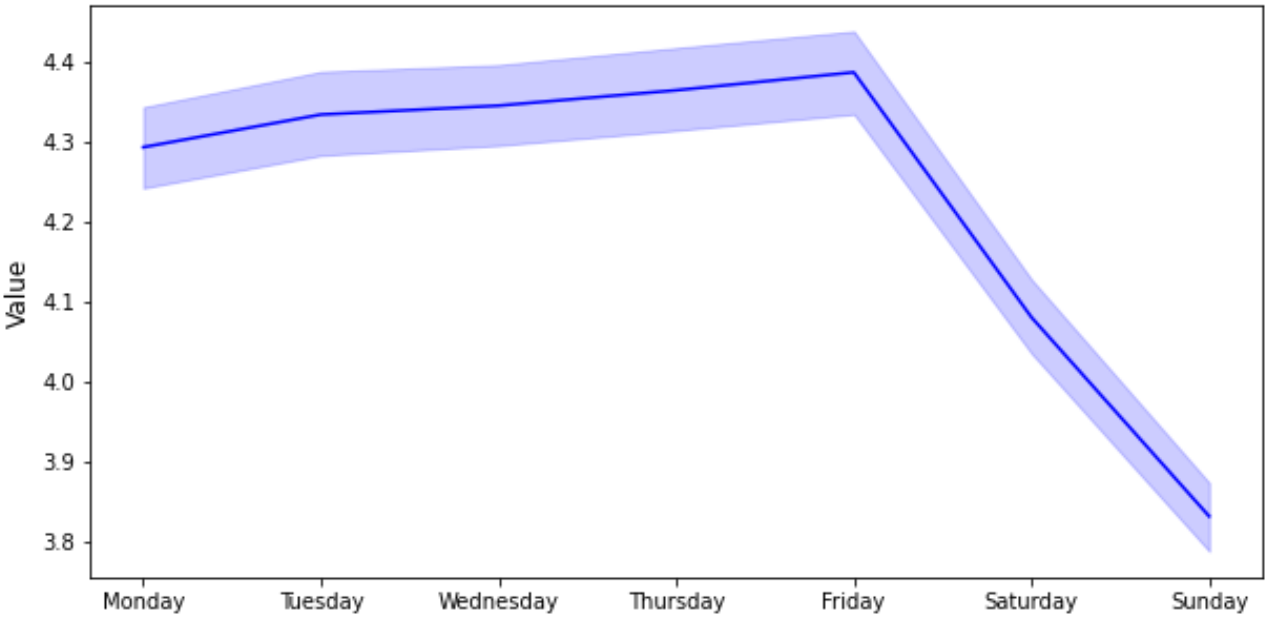

Fig. 8.: andamento settimanale serie storica

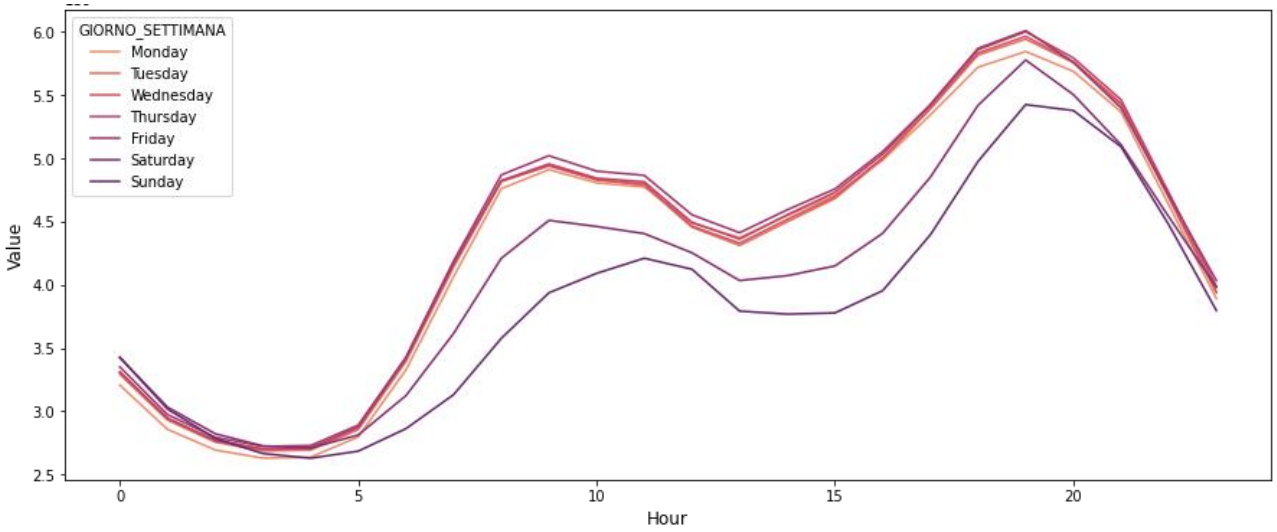

Fig. 9.: andamento ogni ora per giorno della settimana

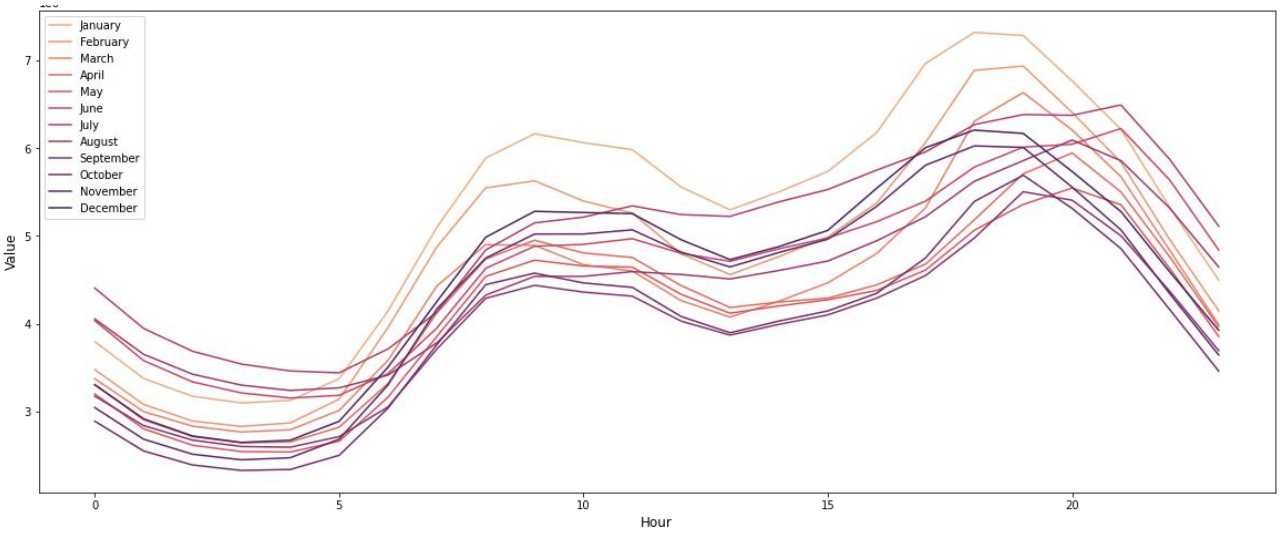

Fig. 10.: andamento ogni ora per mese.

La figura 11 rappresenta un approfondimento sull'influenza che la pandemia da Covid19 ha avuto sui dati. Il lockdown è iniziato il 2020-03-09, da questa data si può osservare una repentina decrescita.

Per rappresentare il periodo della pandemia è stata creata una variabile step "Covid19" che verrà utilizzata al momento dello sviluppo dei modelli.

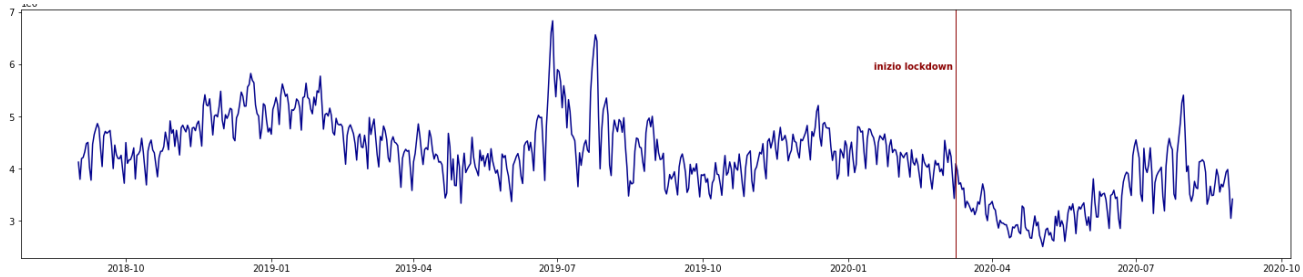

Fig. 11: focus inizio lockdown per pandemia Covid19.

Completate queste altre operazioni si può passare alle fasi di definizione, sviluppo e validazioni dei modelli per un sistema predittivo.

Modelli: Per lo sviluppo di un sistema predittivo sono stati testati modelli ARIMA, UCM, TBATS, PROPHET e LSTM.
La prima operazione è stata quella di definire un dataset di train e uno di validation. Il dataset originario è stato suddiviso nel seguente modo:

- *Dataset di Train*: 14.532 osservazioni, dal 2018-09-01 00:00 al 2020-04-30 23:00:00, pari all'83% del dataset iniziale.
- *Dataset di Validation*: 2.952 osservazioni, dal 2020-05-01 00:00 al 2020-08-31 23:00, pari all'17% del dataset iniziale.

È stato scelto di utilizzare gli ultimi quattro mesi della serie storica per permettere ai modelli di non validare i dati troppo influenzati dalla pandemia da Covid19. I modelli così potranno avere a disposizioni i primi mesi del lockdown partito a marzo 2020 per apprenderne l'andamento e cercare di restituire delle previsioni consone al periodo.

ARIMA: dopo aver suddiviso il dataset originario in train e validation si è iniziato a sviluppare la prima tipologia di modelli, quelli ARIMA. Uno dei requisiti per i modelli ARIMA è che le serie temporali siano stazionarie. Per trasformare una serie non stazionaria in una stazionaria è possibile effettuare trasformazioni logaritmiche (per stabilizzarne la varianza) o differenze (per stabilizzarne la media). Prima di effettuare differenze o trasformazioni logaritmiche ci si è forniti dell'utilizzo dei test Augmented Dickey-Fuller (*ADF*) e Kwiatkowski-Phillips-Schmidt-Shin (*KPSS*) per verificare se la serie storica sia stazionaria oppure no. Con un livello di significatività dello 0,95 , la serie risulta essere stazionaria secondo il test ADF ma non per quello KPSS (Tabella 1.), per questo motivo si è deciso di applicare una differenza stagionale alla serie e successivamente a questa azione la serie risulta essere stazionaria (Tabella 2.).

```
Results of Dickey-Fuller Test:
Test Statistic           -5.810651e+00
p-value                   4.409576e-07
Lags Used                 4.200000e+01
Critical Value (1%)      -3.430800e+00
Critical Value (5%)      -2.861739e+00
Critical Value (10%)     -2.566876e+00

Results of KPSS Test:
Test Statistic            5.779102
p-value                   0.010000
Lags Used                47.000000
Critical Value (10%)      0.347000
Critical Value (5%)       0.463000
Critical Value (2.5%)     0.574000
Critical Value (1%)       0.739000
```

Tabella 1: in alto test iniziale ADF, in basso test iniziale KPSS.

Applicando una differenza stagionale la serie storica risulterà stazionaria come è possibile evincerlo dalla tabella 2. che rappresenta i risultati presenti nei successivi test ADF e KPSS.

```
Results of Dickey-Fuller Test:
Test Statistic           -5.810651e+00
p-value                   4.409576e-07
Lags Used                 4.200000e+01
Critical Value (1%)      -3.430800e+00
Critical Value (5%)      -2.861739e+00
Critical Value (10%)     -2.566876e+00

Results of KPSS Test:
Test Statistic            0.003894
p-value                   0.100000
Lags Used                47.000000
Critical Value (10%)      0.347000
Critical Value (5%)       0.463000
Critical Value (2.5%)     0.574000
Critical Value (1%)       0.739000
```

Tabella 2: in alto test iniziale ADF, in basso test iniziale KPSS dopo aver applicato una differenza stagionale.

Invece, dal correlogramma PACF successivo alla differenza stagionale (Figura 12.) possiamo intuire come un buon modello possa essere uno che abbia come parametri p=1 o p=2.

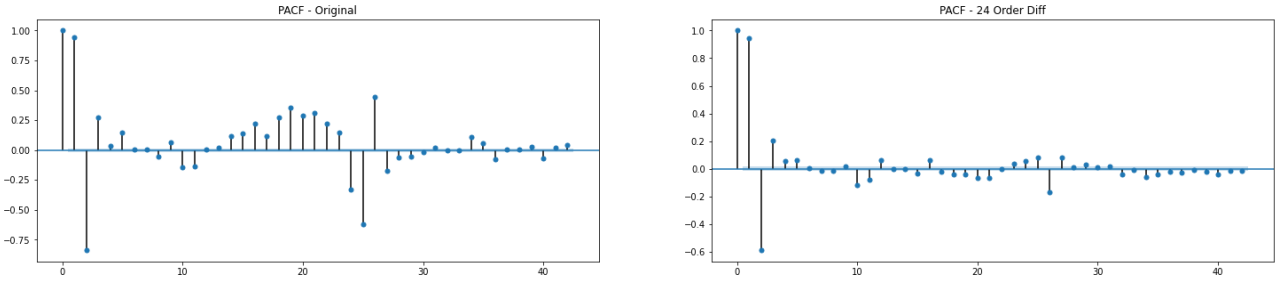

Fig. 12: a sinistra PACF serie storica originale, a destra PACF dopo differenza stagionale della serie storica.

Per determinare i migliori valori per i parametri non stagionali *p* e *q*, è stato utilizzato un approccio *grid search* dove sono stati testati sia i modelli con i dati originali (Tabella 3.) che quelli dopo una trasformazione logaritmica (Tabella 4.).

MODELLO	MAE Train	MAE Validation	AIC
SARIMA(0,0,0)(1,1,1)24	276.458,0	716.203,0	417.218
SARIMA(0,0,1)(1,1,1)24	157.702,5	710.287,1	406.377
SARIMA(0,0,2)(1,1,1)24	110.341,7	707.535,0	400.669
SARIMA(1,0,0)(1,1,1)24	84.912,7	718.818,9	383.424
SARIMA(1,0,1)(1,1,1)24	69.836,8	711.543,9	378.389
SARIMA(1,0,2)(1,1,1)24	66.466,6	710.414,0	377.501
SARIMA(2,0,0)(1,1,1)24	67.167,6	713.421,9	377.707
SARIMA(2,0,1)(1,1,1)24	65.770,2	711.881,0	377.324
SARIMA(2,0,2)(1,1,1)24	65.584,2	711.547,3	377.291

Tabella 3: performance modelli SARIMA con dati originali al variare dei parametri p e q.

MODELLO	MAE Train	MAE Validation	AIC
SARIMA(0,0,0)(1,1,1)24	274.785,7	726.388,8	-29.681
SARIMA(0,0,1)(1,1,1)24	152.181,4	726.320,8	-46.611
SARIMA(0,0,2)(1,1,1)24	104.576,2	726.938,0	-57.133
SARIMA(1,0,0)(1,1,1)24	82.676,4	726.202,6	-63.294
SARIMA(1,0,1)(1,1,1)24	66.682,4	727.167,8	-68.421
SARIMA(1,0,2)(1,1,1)24	63.291,2	727.247,9	-69.288
SARIMA(2,0,0)(1,1,1)24	64.429,8	726.196,4	-68.910
SARIMA(2,0,1)(1,1,1)24	62.807,1	726.737,0	-69.326
SARIMA(2,0,2)(1,1,1)24	62.564,1	726.840,6	-69.379

Tabella 4: performance modelli SARIMA con dati log-trasformati al variare dei parametri p e q.

Come è possibile evincere dalle tabelle 3. e 4. i valori di MAE non differiscono di molto tra i dati originali e quelli con trasformazione logaritmica. Viene così scelto di non applicare nessuna trasformazione logaritmica e di selezionare come miglior modello in termini di AIC e MAE il SARIMA(2, 0, 2)(1, 1, 1)24.

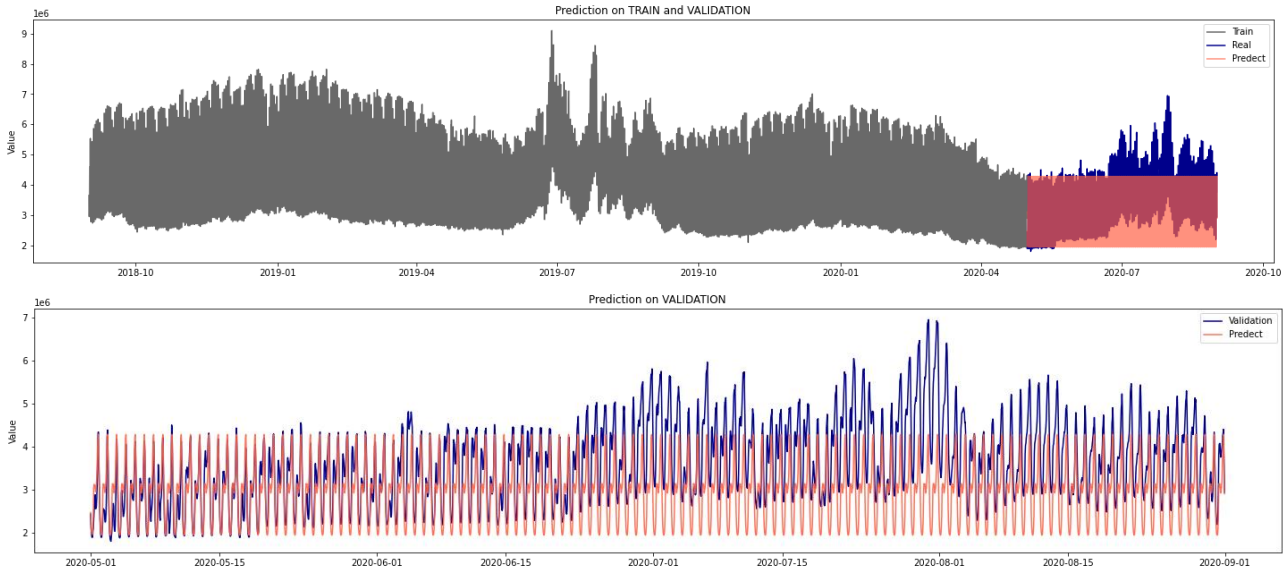

Fig. 13: modello SARIMA(2, 0, 2)(1, 1, 1)24.

Come è possibile osservare all'interno delle figura 13., questo modello non è molto soddisfacente, poiché durante la previsione dei dati di validazione riesce a cogliere la stagionalità giornaliera ma non quella settimanale e annuale. Per ovviare a questo problema di stagionalità multipla, si è deciso di utilizzare un modello SARIMAX.

Vengono inserite delle variabili stagionali costruite tramite serie di Fourier con periodo 168 (settimanale) e 8.766 (annuale). Per scegliere i parametri più performanti viene utilizzato sempre un modello grid search. Dopo questo approccio si è deciso di utilizzare il modello SARIMAX(2,0,2)(1,1,1)24 con 10 armoniche per la

stagionalità settimanale e 5 per quella annuale. Inoltre, vengono effettuate delle prove aggiungendo anche la variabile step 'Covid19' precedentemente costruita senza rilevare miglioramenti.

Si decide quindi di non utilizzarla. Con questo modello rileviamo un MAE sul dataset di train pari a *308.301,8* e un MAE sul dataset di validation pari a *334.117,9*. Osservando la previsione sul dataset di validation, possiamo notare come oltre la stagionalità giornaliera, adesso, il modello segue anche la stagionalità settimanale e annuale (Figura 14. e Figura 15.).

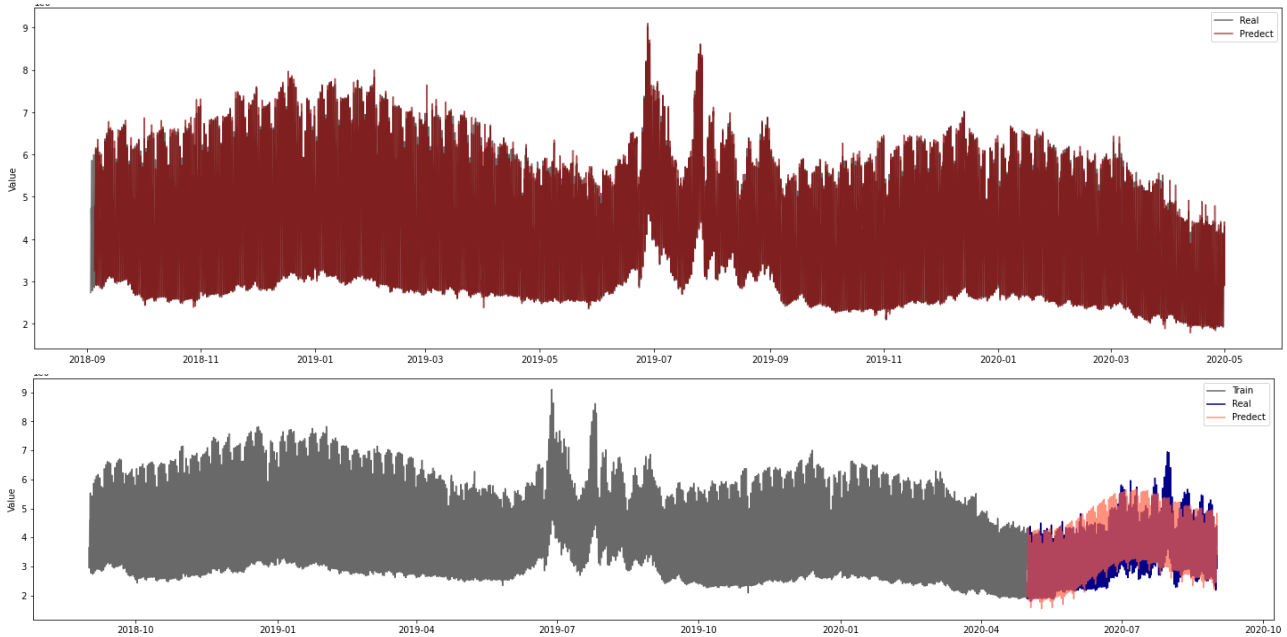

Fig. 14: in alto modello SARIMAX(2, 0, 2)(1, 1, 1)24 confronto dati reali e predetti sul dataset di train e in basso confronto tra i dati reali e quelli predetti sul dataset di validation.

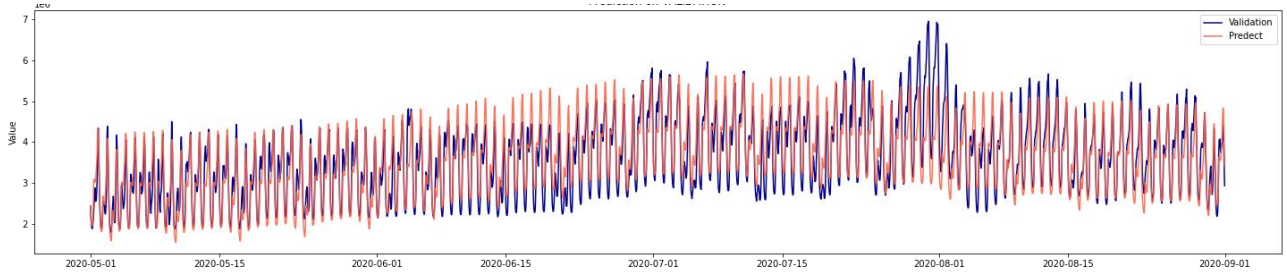

Fig. 15: modello SARIMAX(2,0,2)(1,1,1)24, focus dataset di validation.

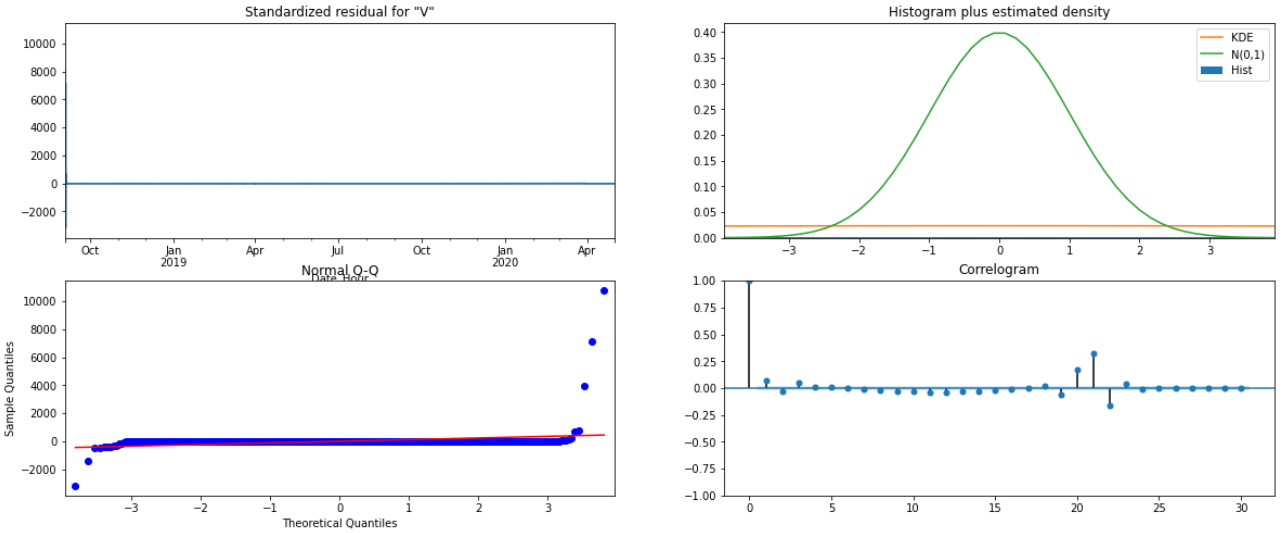

Fig. 16: analisi residui modello SARIMAX(2,0,2)(1,1,1)24.

Dopo aver completato queste operazioni bisognerà validare il modello andando ad osservare le caratteristiche dei residui generati. Per fare ciò, sfruttiamo una funzione presente all'interno della libreria statsmodels di Python. Come è possibile osservare nella figura 16. i residui risultano essere incorrelati e la distribuzione degli stessi risulta approssimativamente normale. Nel complesso, il modello può essere validato.

UCM Oltre ai modelli SARIMA sono stati sviluppati e testati anche i modelli UCM. L'approccio metodologico è stato molto simile a quello utilizzato per validare i modelli SARIMA. È stato suddiviso il dataset originale in train e validation, inserendo nel dataset di validation gli ultimi 4 mesi. Sono state incluse le componenti di stagionalità e di level-trend. Per determinare il level-trend migliore è stato utilizzato un approccio grid search. Dai dati presenti nella tabella 5. riusciamo ad individuare nel random walk con drift il miglior tipo di level-trend per valori di MAE nel dataset di train e di validation. Nella tabella 6. sono presenti i dati con la presenza della variabile step "Covid19" dove però non sono presenti miglioramenti.
La stagionalità giornaliera è rappresentata tramite variabile dummy, mentre quella settimanale tramite 15 serie di Fourier.
Nelle figure 17. e 18. è possibile osservare le previsioni sui dati di train e di validation. Questo modello produce un MAE sul train di *102.623* e *743.459* il MAE di validation.

LEVEL-TREND	MAE Train	MAE Validation
RWALK	102.649	752.062
DCONSTANT	1.023.576	3.645.172
NTREND	1.023.536	3.645.233
LLEVEL	103.983	752.009
LLDTREND	103.958	742.714
RWDRIFT	102.623	743.459
LLTREND	105.020	13.418.369
STREND	105.031	13.418.508
RTREND	103.952	13.096.502

Tabella 5: performance level-trend modelli UCM con dati originali senza variabile covid19.

LEVEL-TREND	MAE Train	MAE Validation

RWALK	102.648	752.062
LLEVEL	103.982	752.009
LLDTREND	103.957	742.704
RWDRIFT	102.622	743.449

Tabella 6: performance level-trend modelli UCM con variabile covid19.

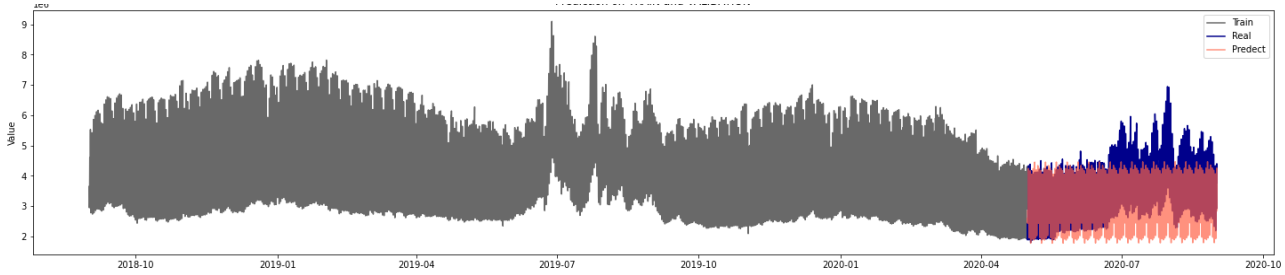

Fig. 17: modello UCM random walk con drift con 15 serie di Fourier per stagionalità settimanale confronto dati reali e predetti sul dataset di validation.

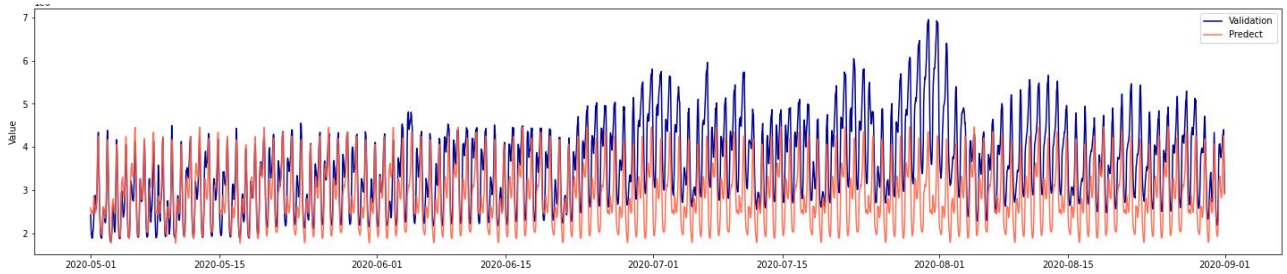

Fig. 18: focus su previsione dati di validation modello UCM random walk con drift con 15 serie di Fourier per stagionalità settimanale.

PROPHET Prophet è una libreria open source, sviluppata da Facebook che ha come scopo quello di prevedere serie temporali univariate. La stessa, è progettata per essere facile e automatica e implementa un modello additivo che supporta trend, stagionalità e festività.

Questo modello è stato sviluppato e validato con stagionalità giornaliera, settimanale e annuale. Le armoniche adoperate per ogni stagionalità sono:

- 3 armoniche per la stagionalità settimanale;
- 4 armoniche per la stagionalità giornaliera;
- 10 armoniche per la stagionalità annuale.

Alla fine questo modello presenta un MAE sul Train di *366.467,4* e un MAE sul Validation di *545.685,0* .

All'interno della figura 19 è possibile osservare i grafici che rappresentano le previsioni sul dataset di Train e in quello di Validation.

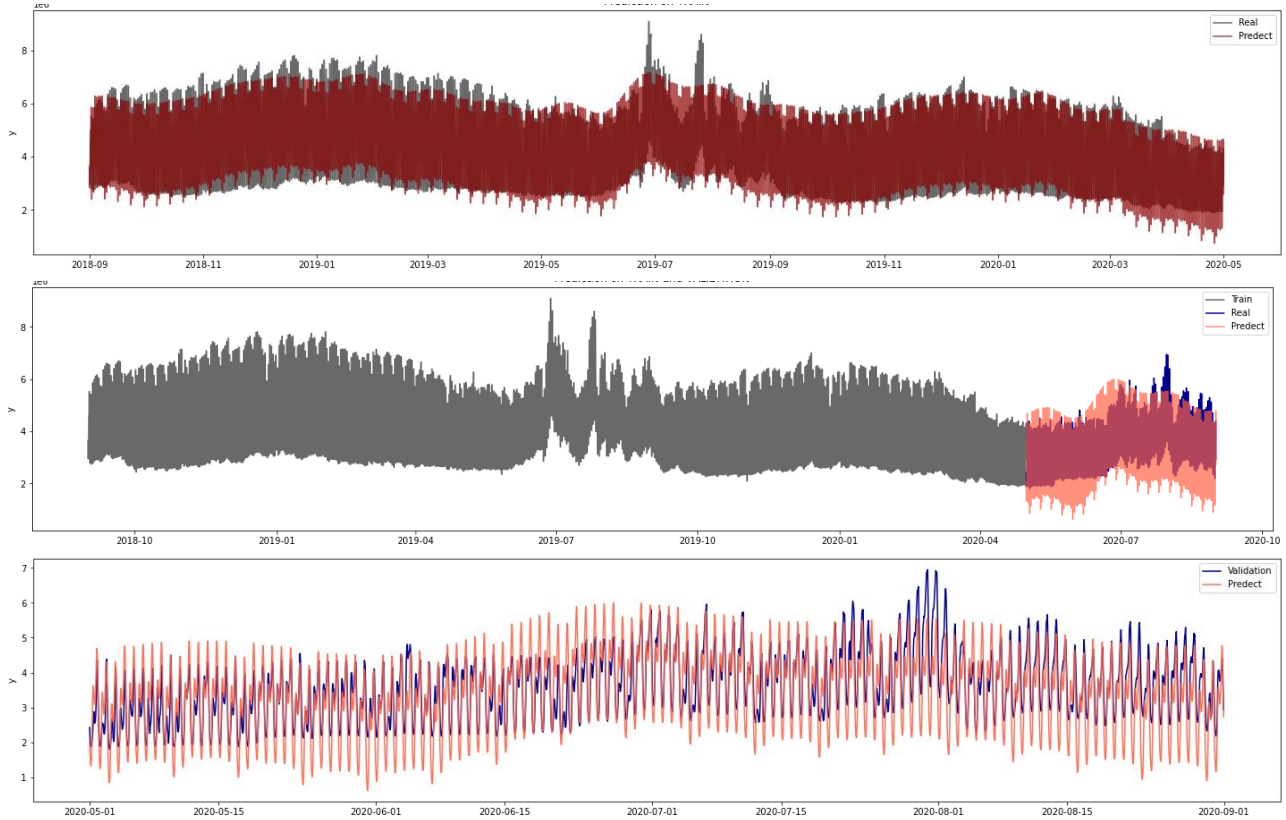

Fig. 19: modello PROPHET, prima immagine previsioni sul Train set, seconda immagine previsioni sul validation set e terza immagine focus su previsioni validation set.

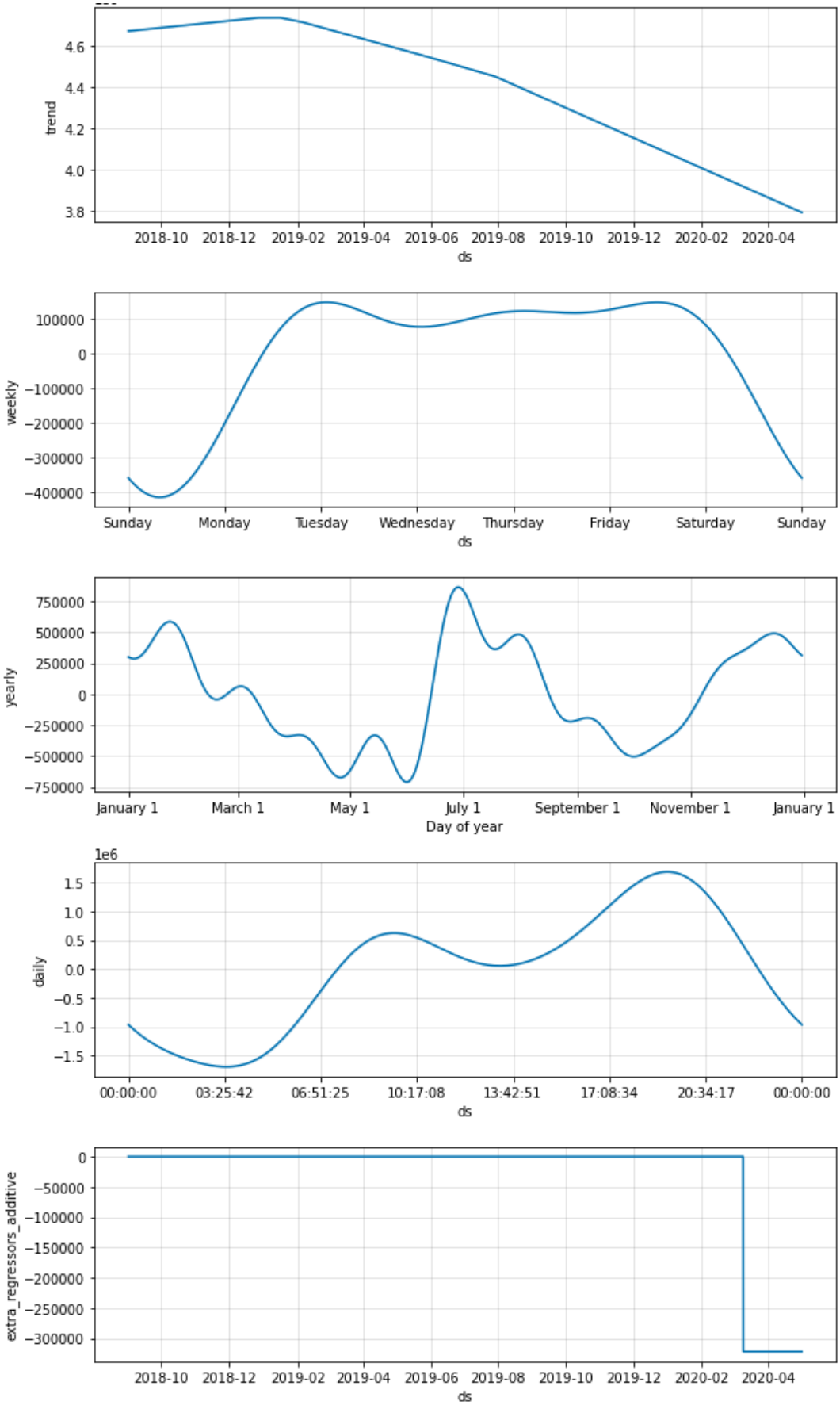

Fig. 20: Componenti del modello PROPHET.

Dalla figura 20 possiamo notare la decomposizione delle diverse componenti utilizzate nel modello Prophet, ovvero:

- *Trend;*
- *Stagionalità giornaliera;*
- *Stagionalità settimanale;*
- *Stagionalità annuale;*
- *Regressore esterno Covid19.*

Inoltre, PROPHET come input presenta alcuni iperparametri che possono essere ottimizzati. I valori che portano ad un MAE inferiore sono, changepoint_prior_scale uguale a *0,01* e seasonality_prior_scale uguale a *0,01* (tabella 7) che verranno utilizzati per fare le previsioni.

changepoint_prior_scale	seasonality_prior_scale	mae
0.001	0.01	6,16E+11
0.001	0.10	6,23E+11
0.001	1.00	6,20E+11
0.001	10.00	6,24E+11
0.010	0.01	5,95E+11
0.010	0.10	6,05E+11
0.010	1.00	6,10E+11
0.010	10.00	6,12E+11
0.100	0.01	6,19E+11
0.100	0.10	1,01E+12
0.100	1.00	9,81E+11
0.100	10.00	8,21E+11
0.500	0.01	1,12E+12
0.500	0.10	2,75E+12
0.500	1.00	3,36E+12
0.500	10.00	3,06E+12

Tabella 7: iperparametri Prophet.

TBATS I modelli TBATS (Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components) sono modelli che utilizzano stagionalità trigonometrica, trasformazione Box-Cox, errori ARMA e componenti Trend e Stagionale che compongono l'acronimo che rappresenta il nome di questi modelli. Il modello più performante in termini di MAE è stato quello con trasformazione Box-Cox, modellazione degli errori tramite processo ARMA e stagionalità multipla (giornaliere, settimanale e annuale). Quest'ultimo modello ci fornisce come output un MAE sul train set uguale a *100.143,1* e MAE sul validation set uguale a *620.078,5* . La figura 21 mostra i grafici sulle previsioni sul dataset di Train e di Validation.

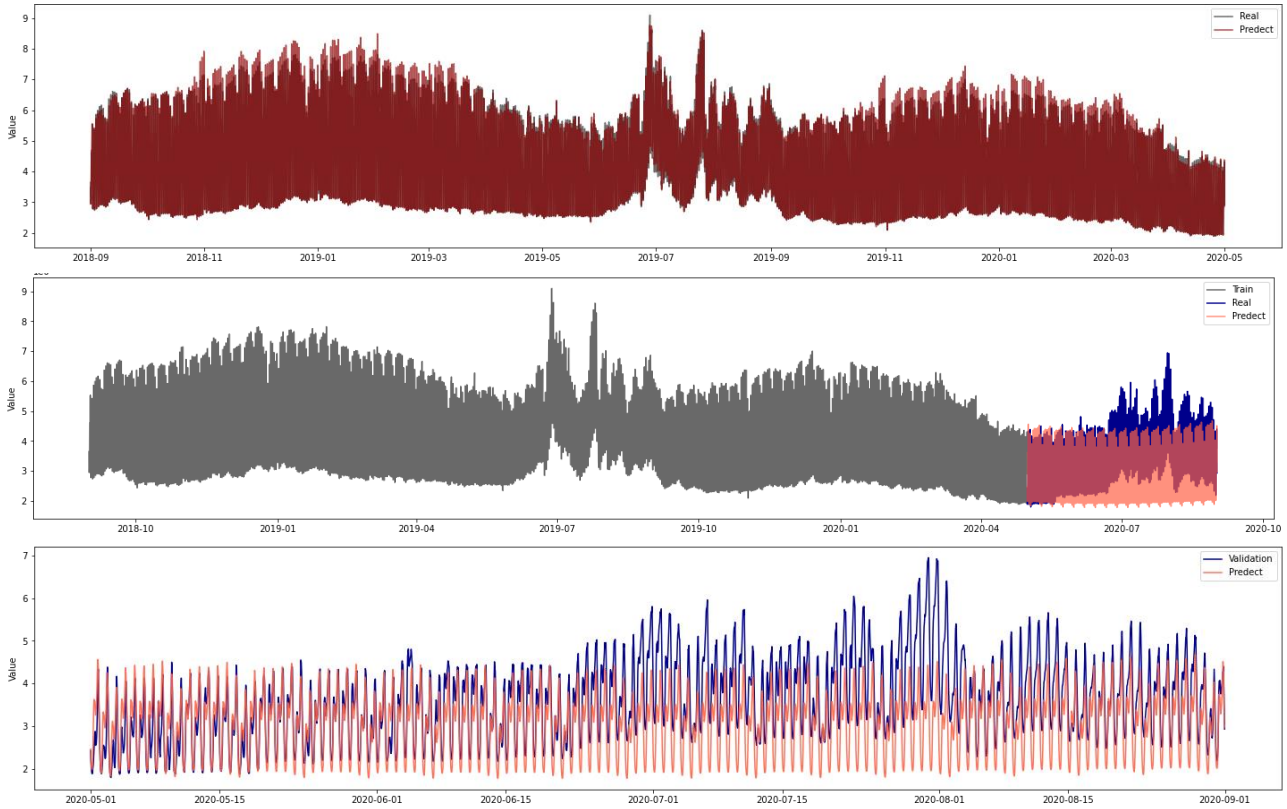

Fig 21: modello TBATS, prima immagine previsioni sul Train set, seconda immagine previsioni sul validation set e terza immagine focus su previsioni validation set.

LSTM Il modello non lineare scelto sono le reti neurali *Long short-term memory*. Per questi modelli sono stati scalati i dati all'interno dell'intervallo tra 0 e 1. Inoltre, si è scelto una implementazione stateless, ovvero, la rete neurale non tiene conto di quello che è successo tra un batch e un altro. Il dataset di Validation utilizzato presenta soltanto 61 giorni.

Il modello scelto presenta la seguente architettura:

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 100)	40800
leaky_re_lu_1 (LeakyReLu)	(None, 100)	0
dropout_1 (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 744)	75144

Total params: 115,944
Trainable params: 115,944
Non-trainable params: 0

Tabella 8: Architettura LSTM

Il MAE sul Train set di questo modello è uguale a *1.105.124* e MAE sul validation set pari a *695.363* .

Nella figura 22 è possibile osservare i grafici sulle previsioni sul dataset di Train e Validation.

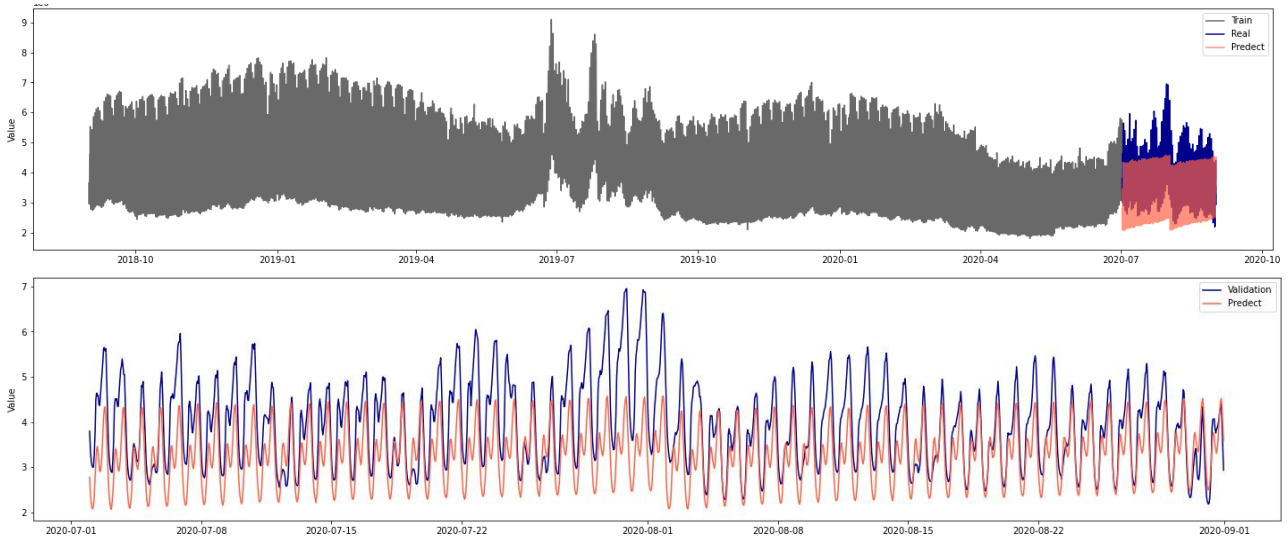

Fig. 22: modello LSTM, in alto previsioni sul validation set, in basso focus su previsioni validation set.

Risultati. Osservando i risultati dei vari modelli, per ogni tipologia è stato selezionato quello con il valore in output di MAE migliore. Una volta scelti i vari modelli si è passati alla previsione della colonna VALORE per il range temporale che va dal 2020-09-01 al 2020-10-31, che corrisponde all'obiettivo di questo progetto. La figura 23 riporta i grafici di ogni modello con il valore previsto della colonna VALORE.

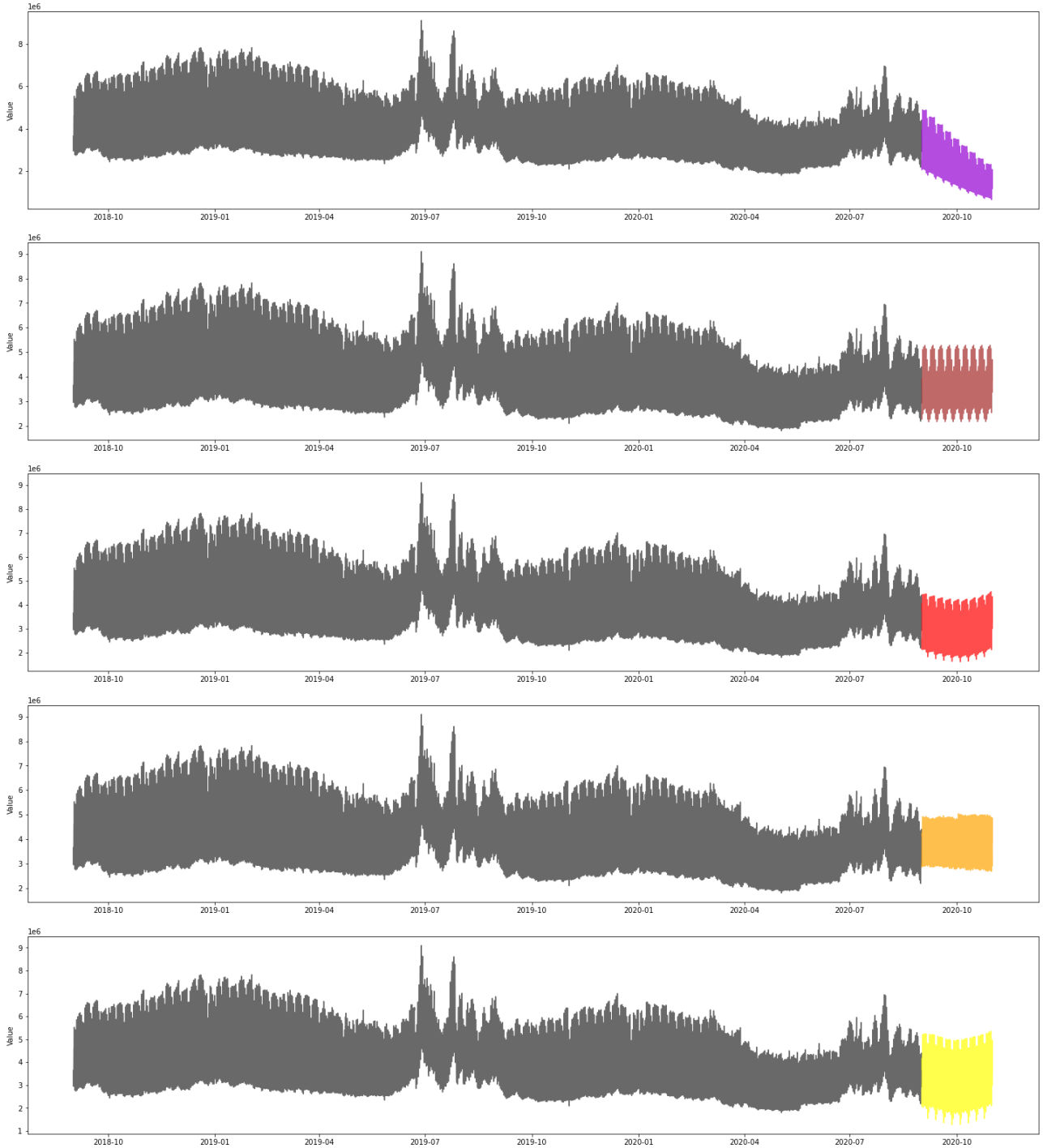

Fig. 23 I modelli utilizzati per la previsione futura, in ordine: TBATS, UCM, SARIMAX, LSTM, PROPHET.

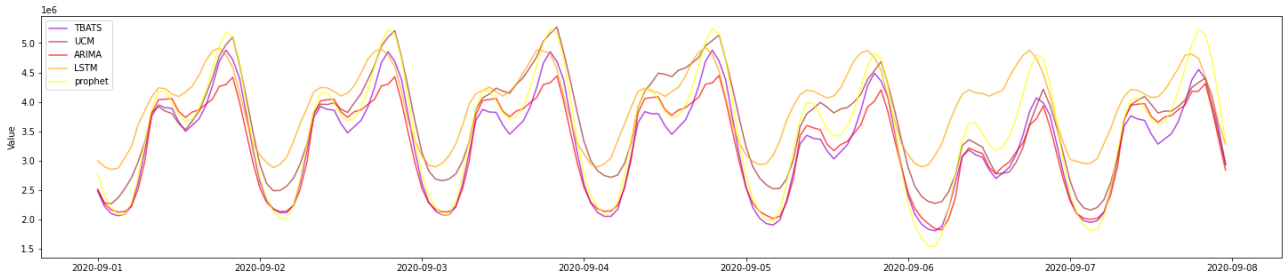

Fig. 24: Focus previsioni sulla prima settimana.

Infine, all'interno della tabella 7 sono presenti tutti i dati sul MAE di Train e sul MAE di Validation dei vari modelli selezionati. Osservando i diversi valori di MAE, si può concludere che il modello SARIMAX(2,0,2)(1,1,1)24 risulta essere il migliore.

Modello	MAE Train	MAE Validation
UCM	102.623,3	743.458,6
SARIMAX(2,0,2)(1,1,1)24	308.301,8	334.117,9
TBATS	100.143,1	620.078,5
PROPHET	366.467,4	545.685,0
LSTM	1.105.124,1	695.363,2

Tabella 7: Modelli più performanti.

Conclusione e possibili sviluppi. Questo progetto ha portato a definire un possibile sistema predittivo dei dati forniti. Alla fine del progetto, il modello SARIMAX(2,0,2)(1,1,1)24 è risultato il più performante. Per ottenere dei risultati migliori occorrerebbe che si venisse a conoscenza della tipologia dei dati forniti. Inoltre, la presenza di dati aggiuntivi precedenti al 2018-09-01 potrebbero essere utili per migliorare le prestazioni dei modelli, poiché, essi riuscirebbero ad assimilare di più nella fase di apprendimento.

Riferimenti bibliografici.

Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras:
https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/
Generate Quick and Accurate Time Series Forecasts using Facebook's Prophet (with Python & R codes):
https://www.analyticsvidhya.com/blog/2018/05/generate-accurate-forecasts-facebook-prophet-python-r/
Time Series Forecasting With Prophet in Python:
https://machinelearningmastery.com/time-series-forecasting-with-prophet-in-python/
Forecasting Time Series Data with Multiple Seasonal Periods:
https://tanzu.vmware.com/content/blog/forecasting-time-series-data-with-multiple-seasonal-periods
Hourly electricity demand forecasting using Fourier analysis with feedback:
https://www.sciencedirect.com/science/article/pii/S2211467X20300778
How to Grid Search SARIMA Hyperparameters for Time Series Forecasting:
https://machinelearningmastery.com/how-to-grid-search-sarima-model-hyperparameters-for-time-series-forecasting-in-python/