UNIVERSITÀ DEGLI STUDI DI MILANO

Master in Data Science for Economics, Business and Finance

Analisi e Forecasting di una Serie Temporale

Tesi di fine Master di:
Claudio PANELLI
Matr. A04422

Anno Accademico 2021/2022

Indice

Introduzione **5**

1 Nozioni Preliminari **7**

1.1 One Step Forecasting 7

1.2 Metriche 8

1.2.1 MSE 8

1.2.2 RMSE 9

1.2.3 MAE 9

2 Analisi Descrittiva **11**

2.1 Descrizione 11

2.2 Rappresentazione Grafica 11

2.3 Conclusioni 12

3 ARIMA **15**

3.1 Modello di Autoregressione - AR(p) 15

3.2 Modello di Moving Average – MA(q) 16

3.3 Stazionarietà 16

3.4 Modelli ARMA e ARIMA 17

3.4.1 ACF e Ricerca del valore q 19

3.4.2 PACF e Ricerca del valore p 20

3.4.3 AIC e BIC 20

3.4.4 Predizione 22

4 Recurrent Neural Network **23**

4.1 Reti Neurali Ricorrenti - Simple RNN 23

4.1.1 Unfolding della rete 24

4.1.2 Training RNNs 26

4.1.3 Vanishing/Exploding Gradient 26

4.2 Rete Neurale Ricorrente e Risultati 27

4.2.1 RNN 27

4.2.2 Dataset di Train/Validation e Test 28

4.2.3 Risultati 29

5 Prophet **31**

5.1 Introduzione 31

5.2 Trend 32

5.2.1 Modelli di saturazione non lineare 32

5.2.2 Trend lineari con change points 33

5.3 Stagionalità 34

5.4 Holiday 35

5.5 Setup di Prophet e Risultati 35

5.5.1 Setup di Prophet 35

5.5.2 Risultati 37

6 Conclusioni **39**

Bibliografia **41**

Introduzione

Lo scopo della tesi è studiare la serie storica rappresentante la media delle ore settimanali lavorative nel settore Manufacturing e applicare tre modelli predittivi: ARIMA, RNN e Prophet.
La Tesina è così suddivisa:

- Nel Capitolo 1 descriviamo gli approcci seguiti per eseguire il forecasting della serie storica e le metriche che useremo per misurare i risultati ottenuti.
- Nel Capitolo 2 descriviamo la serie storica in oggetto: l'eventuale trend, le stagionalità e ne rappresentiamo i valori in un diagramma temporale.
- Nel Capitolo 3 introduciamo il metodo ARIMA e vediamo tale metodologia applicata alla serie storica.
- Nel Capitolo 4 descriviamo le reti neurali ricorsive e proviamo a predirre i valori futuri della serie storica mediante una rete di questo tipo.
- Nel Capitolo 5 descriviamo l'algoritmo Prophet e lo applichiamo alla serie storica.
- Nel Capitolo 6 riassumiamo i risultati ottenuti nell'applicare i modelli predittivi alla serie storica.

Capitolo 1

Nozioni Preliminari

Prima di procedere con la trattazione degli argomenti presentati, descriviamo gli approcci seguiti per eseguire il forecasting della serie storica e le metriche che useremo per misurare i risultati ottenuti.

1.1 One Step Forecasting

Gli approcci che useremo per eseguire il forecasting della serie storica sono: il "One step forecast con re-estimation" e il "One step forecast senza re-estimation".
La differenza tra i due approcci risiede nel fatto che, nel primo caso, rieseguiamo la fase di istruzione del modello dopo ogni predizione, mentre, nel secondo approccio, il modello che useremo per la predizione non cambierà.
Più in dettaglio, il "One step forecast con re-esimation" prevede i seguenti passi:

- istruisco il modello sui T dati del dataset di train;
- predico il primo punto del dataset di test, ovvero quello al tempo $T + 1$;
- memorizzo la predizione per $T+1$ in un appostio array che userò successivamente per calcolare le performance del modello in oggetto;
- al dataset di train di T elementi aggiungo il primo punto del dataset di test, quello al tempo $T + 1$;

- reistruisco il modello sui $T+1$ dati ottenuti;
- utilizzando il nuovo modello, predico il secondo punto del dataset di test, quello al tempo $T+2$ ... e così via;
- continuo fino a terminare i dati del dataset di test.

A differenza del primo approccio, il "One step forecast senza re-estimation" prevede che il modello venga istruito una sola volta sui dati provenienti dal dataset di train e lo si utilizzi, passo dopo passo, per la predizione di tutti i punti del dataset di test.

1.2 Metriche

Le metriche che useremo per valutare le performance dei modelli sono: MSE, $RMSE$ e MAE. Essendo metriche che misurano l'errore commesso, l'obiettivo degli algoritmi, sarà minimizzare tali valori.

1.2.1 MSE

MSE (Mean Square Error): misura la media dei quadrati degli errori, ovvero la differenza quadratica media tra i valori stimati e i valori effettivi.
Formalmente è rappresentato dalla funzione:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \approx E\left((y-\hat{y})^2\right)$$

dove: n è il numero dei punti della serie temporale.

Vantaggi

- essendo una media, il valore dell' MSE è indipendente dal numero dei punti della serie, ovvero l'MSE è una metrica di errore invariante rispetto al numero di campioni;
- MSE è anche una buona stima del valore atteso dell'errore commesso al quadrato.

Svantaggi

- l'unità di misure di MSE non è intuitiva se confrontata con il valore dei punti della serie storica.

1.2.2 RMSE

RMSE (Root Mean Square Error): radice quadrata della media dei quadrati degli errori tra i valori stimati e i valori effettivi.
Formalmente è rappresentato dalla funzione:

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

dove: n è il numero dei punti della serie temporale.

Vantaggi

- il valore dell'RMSE è confrontabile con i valori della series storica; riportiamo la misura di errore sulla stessa scala dei dati originali.

Svantaggi

- bisogna prestare attenzione al fatto che la radice quadrata di un numero maggiore di uno viene ridotta, mentre la radice quadrata di un numero minore di uno viene aumentata.

1.2.3 MAE

MAE (Mean Absolute Error): misura la media degli errori in valore assoluto, ovvero l'errore medio assoluto.
Formalmente è rappresentato dalla funzione:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

dove: n è il numero dei punti della serie temporale.

Vantaggi

- il calcolo del MAE è semplice ed è anche confrontabile con i valori della series storica;
- l'indicatore MAE è robusto in presenza di outliers.

Svantaggi

- la misura è, come l'MSE e l'RMSE, dipendente dalla scala dei dati della serie storica in analisi.

L'obiettivo della tesi è quello di confrontare tre modelli predittivi sulla stessa serie storica, quindi, semplicemente, ci aspettiamo che il modello migliore abbia MSE, RMSE e MAE minimi.

Capitolo 2

Analisi Descrittiva

2.1 Descrizione

La serie presa in considerazione è la Average Weekly Hours of Manufacturing Employees, (https://fred.stlouisfed.org/series/AWHMAN) e rappresenta la media delle ore settimanali lavorative nel settore Manufacturing.
È misurata in ore. La frequenza dei campioni è mensile. Nessuna trasformazione è stata applicata prima di iniziare l'analisi.

2.2 Rappresentazione Grafica

La serie originale contiene i dati che vanno da Gennaio 1959 a Dicembre 2021 per un totale di 758 campionamenti. Per la mia analisi ho selezionato solo i 480 campioni più recenti che vanno da Gennaio 1980 a Dicembre 2019.
Nel grafico 2-1 rappresentiamo i valori della serie storica oggetto della tesi da Gennaio 1980 a Dicembre 2019.
Nel grafico 2-2 rappresentiamo la stessa serie in cui però distinguiamo i punti del train e del test dataset.
Nota: i dati del train rappresentano l'80% del dataset originale.

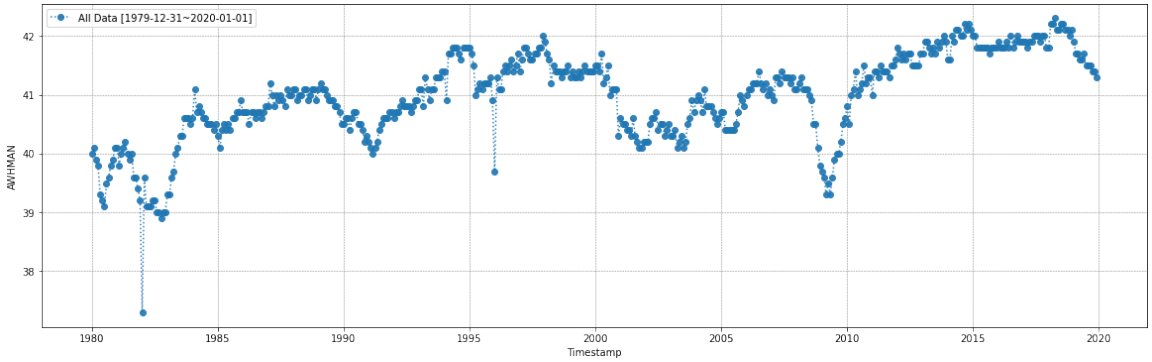

Figura 2-1: Time Series - All Data

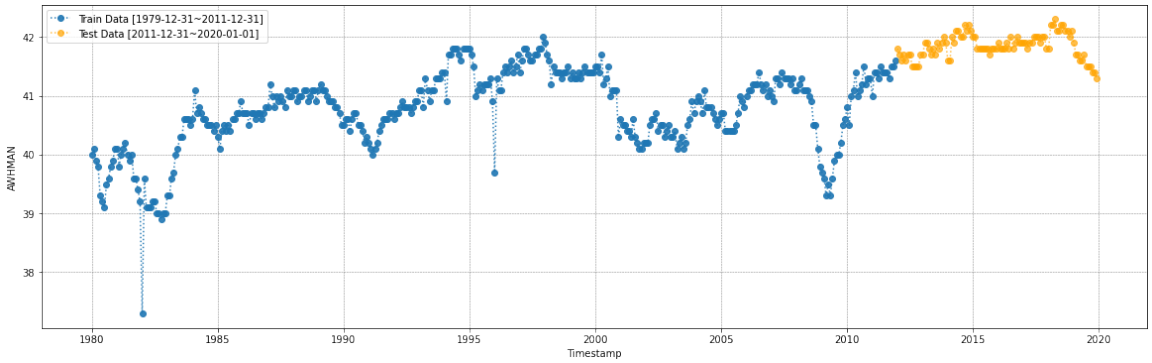

Figura 2-2: Time Series - Train e Test Data

2.3 Conclusioni

Possiamo osservare dai grafici una sostanziale mancanza di trend; ci aspettiamo dunque che la serie sia stazionaria. Per quanto riguarda la stagionalità, non siamo in grado di osservarne con certezza, nessuna.

Proviamo dunque a decomporre la serie in trend e stagionalità mediante la funzione `seasonal_decompose` del pacchetto `statsmodels.tsa.seasonal`. Per motivi di spazio, non approfondiamo lo studio della funzione `seasonal_decompose` nella corrente tesi e rimandiamo alla pagina ufficiale eventuali approfondimenti [1].

Grazie alla decomposizione possiamo affermare che la serie storica non possiede nè un trend, nè una stagionalità.

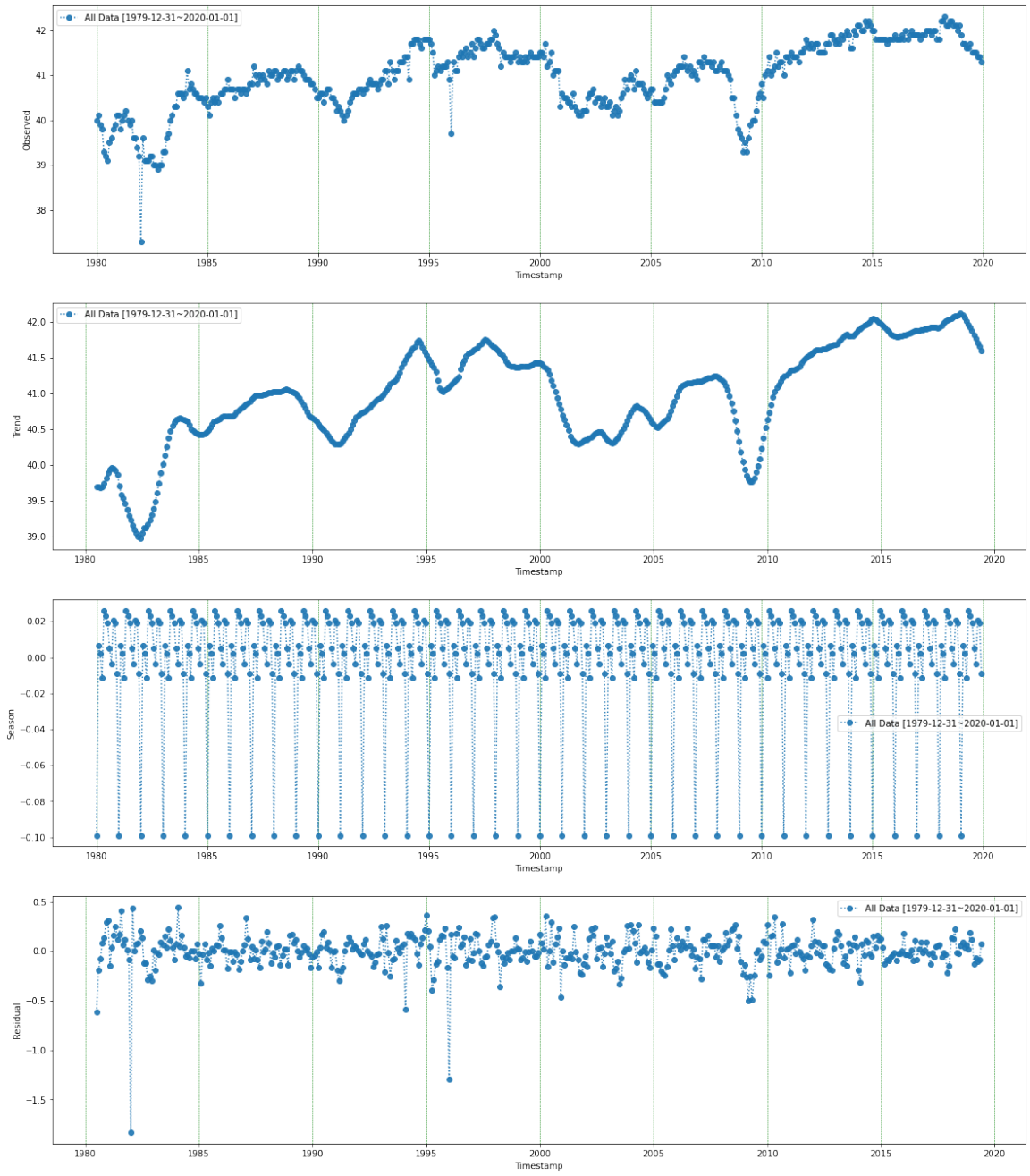

Figura 2-3: Time Series - Trend, Seasonality e Residual

Capitolo 3

ARIMA

In questo capitolo descriviamo i modelli di autoregressione (AR), di moving average (MA) e la combinazione dei due modelli, ovvero il modello ARMA/ARIMA.
Infine, vediamo l'applicazione del modello ARIMA alla serie temporale in oggetto e i risultati ottenuti.

3.1 Modello di Autoregressione - AR(p)

Definiamo un modello di autoregressione di ordine p, o modello $AR(p)$, come un modello di regressione multi lineare dove, come input, abbiamo i p valori del passato della serie temporale $(y_{t-1}, y_{t-2}, \ldots, y_{t-p})$.
Formalmente, possiamo esprimere un modello $AR(p)$, come:

$$y_t = b + \varphi_1 y_{t-1} + ... + \varphi_p y_{t-p} + \varepsilon_t \tag{3.1}$$

$$\varepsilon_t \sim N\left(0, \delta^2\right)$$

Dove, il termine ε_t rappresenta il rumore al tempo t.
L'equazione 3.1 esprime la dipendenza lineare del valore corrente y con i valori passati della serie temporale $(y_{t-1}, y_{t-2}, \ldots, y_{t-p})$ sommati all'errore incontrato al tempo t. Tale errore è rappresentato da un valore ricavato da una distribuzione normale di media 0 e deviazione standard δ^2.

3.2 Modello di Moving Average – MA(q)

Un modello di moving average è espresso dalla seguente equazione:

$$y_t = b + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

I modelli di moving average sono simili ai precedenti, gli autoregessivi, in quanto entrambi esprimono il valore corrente della serie temporale y_t come linearmente dipendente dai valori del passato. Nel caso del modello moving average, però, il valore corrente della serie temporale y_t dipende linearmente dai valori degli errori passati, $(\varepsilon_{t-1}, ..., \varepsilon_{t-q})$, più quello corrente ε_t.

3.3 Stazionarietà

Prima di introdurre i modelli ARMA e ARIMA definiamo il concetto di serie temporale stazionaria.
Una serie temporale è detta stazionaria quando la distribuzione dei dati della serie non cambia nel tempo, ovvero quando la media e la varianza dei valori della serie è costante.
Dimostriamo la stazionarietà di una serie mediante il test statistico Augmented Dickey-Fuller (ADF). Tale test ci dice se la serie ha una radice unitaria, ovvero se la serie possiede un trend stocastico.
L'equazione del test è:

$$\Delta y_t = c + \beta * y_{t-1} + u_t$$

Le ipotesi:

- H_0 : L'ipotesi nulla è che beta non è significativo e quindi la serie è non stazionaria.
- H_1 : L'ipotesi alternativa è che beta sia significativo e quindi la serie è stazionaria.

Interpretiamo il risultato del test ADF usando il valore p_value del test.

- se il $p\_value > soglia\_critica$ allora non rigettiamo l'ipotesi nulla di non stazionarietà: la serie è non stazionaria.

- se il $p\_value < soglia\_critica$ allora possiamo rigettare l'ipotesi nulla di non stazionarietà: la serie è stazionaria.

quindi (usando una soglia critica del 5%):

- $p\_value > 0.05$: la serie è non stazionaria.

- $p\_value <= 0.05$: la serie è stazionaria.

Calcoliamo il Test di Augmented Dickey-Fuller per la serie temporale oggetto della tesi (vedi figura: 3-1). Il risultato, $p\_value = 0.027512 < 0.05$, assicura la stazionarietà della serie.
Possiamo quindi applicare i metodi ARMA/ARIMA direttamente alla serie senza doverla differenziare.

```
ADF Statistic: -3.087246
p-value: 0.027512
Critical Values:
        1%: -3.448
        5%: -2.869
        10%: -2.571
Time Series is Stationary
```

Figura 3-1: Augmented Dickey-Fuller (ADF)

3.4 Modelli ARMA e ARIMA

Definiamo i modelli ARMA e ARIMA, come combinazione dei modelli studiati in precedenza.
Un modello ARMA è costruito sommando un modello di autoregressione di ordine p con un moving average di ordine q:

$$y_t = b + \varphi_1 y_{t-1} + ... + \varphi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Usiamo, quindi, questo modello se siamo sicuri che ogni punto della serie temporale y_t sia linearmente correlato con i precedenti valori della serie $y_{t-1}, y_{t-2}, ..., y_{t-p}$ e con gli errori commessi $\varepsilon_{t-1}, \varepsilon_{t-2}, ..., \varepsilon_{t-q}$.
Introduciamo ora i modelli ARIMA.
Definiamo innanzitutto l'operazione di differenziazione come:

$$\Delta y_t = y_t - y_{t-1}$$

Dove, ad ogni dato della serie storica, sottraiamo il valore precedente.
La serie storica ottenuta Δy_t rappresenta una nuova serie storica che, rispetto alle originali, è senza trend, ovvero, mediante la differenziazione, otteniamo una serie temporale stazionaria le cui componenti statistiche (media e varianza) rimangono costanti nel tempo. Ciò si rende utile per istruire correttamente un modello ARMA in quanto ogni finestra temporale della serie storica sarà simile alle altre.
Definiamo un modello $ARIMA(p, d, q)$ come un $ARMA(p, q)$ la cui serie temporale è stata differenziata d volte con lo scopo di renderla stazionaria. In altri termini, il modello $ARIMA(p, d, q)$ è un modello $ARMA(p, q)$ in cui abbiamo integrato l'operazione di differenziazione d volte.
Osserviamo che:

- $ARIMA(p, 0, 0)$ coincide con $ARMA(p, 0)$ e un $AR(p)$
- $ARIMA(0, 0, q)$ coincide con $ARMA(0, q)$ e un $MA(q)$
- $ARIMA(0, d, 0)$ coincide con $I(d)$
- $ARIMA(0, 1, 0)$ coincide con $I(1)$ ovvero un Random Walk

Vediamo ora le tecniche utilizzate per selezionare i restanti parametri del modello ARIMA: p e q e le metriche per la valutazione di tali modelli.

3.4.1 ACF e Ricerca del valore q

La funzione di Auto Correlazione (ACF) è una funzione matematica che ci aiuta a capire la dipendenza che hanno i valori correnti della serie temporale con i dati dei periodi precedenti, ovvero, il valore della funzione ACF, misura il grado di dipendenza che i dati attuali mostrano con quelli passati.
Chiamiamo la singola unità di distanza tra i dati in oggetto e quelli del passato con il termine *lag*.
Rappresentiamo la funzione ACF con un grafico che riporta sull'asse delle ascisse la distanza in *lag* tra i dati in oggetto e quelli del passato e sull'asse delle ordinate il valore della funzione ACF su tali distanze.
Il valore q della componente di moving average è equivalente all'ultimo valore di *lag* per cui il valore della ACF è ancora significativo, ovvero una serie temporale è rappresentabile con un modello di moving average di ordine q presenta un valore significativo di Auto Correlazione a distanza non maggiore di q.
Calcoliamo il diagramma ACF per la serie temporale in oggetto. Il grafico indica, come scelta del valore di q, un valore elevato (vedi figura: 3-2). Ciò implicherebbe però la creazione di un modello complesso.
Vedremo nella sezione 3.4.3 un modo migliore per la scelta di tale parametro.

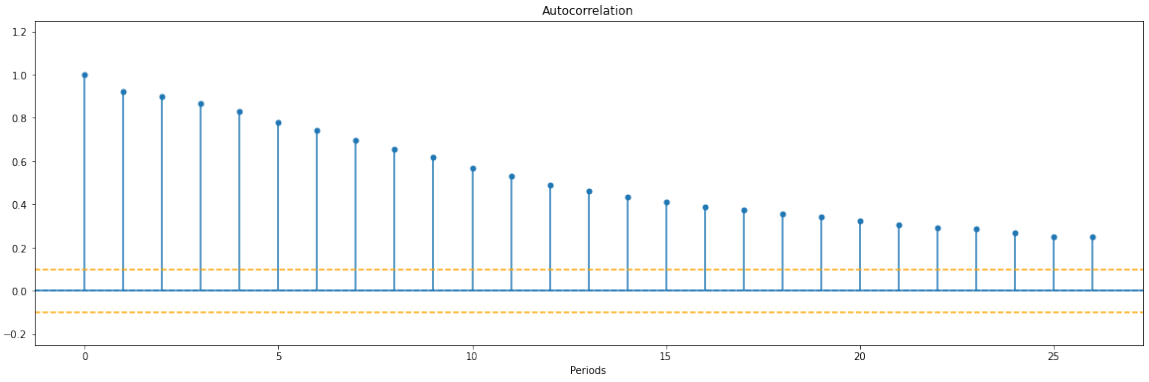

Figura 3-2: Diagramma ACF

3.4.2 PACF e Ricerca del valore p

La funzione di Auto Correlazione Parziale $PACF$ misura il grado di dipendenza che i dati in oggetto mostrano con i dati dei periodi precedenti al netto dei periodi intermedi.

Come fatto per la funzione ACF, rappresentiamo la $PACF$ con un grafico che riporta sull'asse delle ascisse la distanza in lag tra i dati in oggetto e quelli del passato e sull'asse delle ordinate il valore della funzione $PACF$ su tali distanze.

E' dimostrabile che una serie temporale rappresentabile con un modello di auto regressione di ordine p presenta un valore significativo di Auto Correlazione Parziale a distanza non maggiore di p.

Calcoliamo il diagramma PACF per la serie storica. Il diagramma indica che, un buon valore per p, dovrebbe poter essere 1 oppure 2 (vedi figura: 3-3).

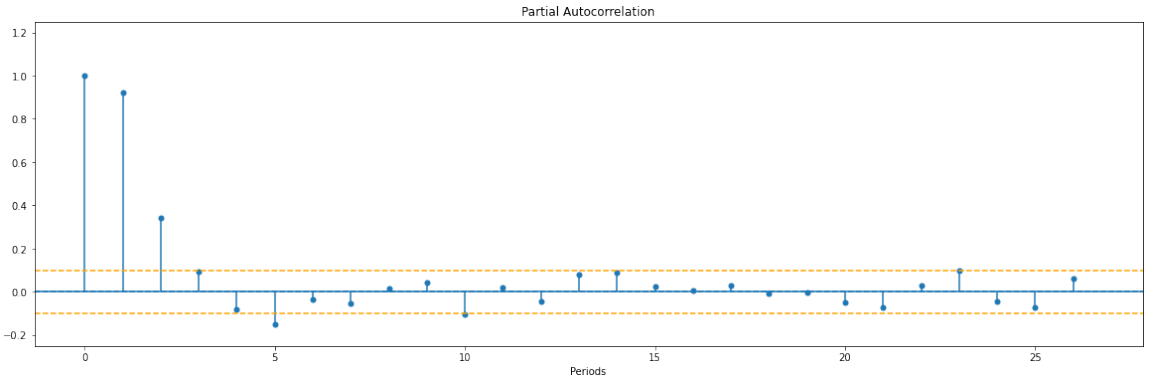

Figura 3-3: Diagramma PACF

3.4.3 AIC e BIC

I diagrammi ACF e PACF ci danno un idea iniziale e a volte approssimativa dei valori ottimali da scegliere per le componenti $AR(p)$ e $MA(q)$ del modello $ARIMA$. Per la scelta ottimale di tali parametri, p e q, ci affidiamo alle metriche AIC (Akaike information criterion) e BIC (Bayesian information criterion).

Il criterio dell'informazione Akaike (AIC) e il criterio dell'informazione Bayesiano (BIC) misurano le prestazioni di un modello tenendo conto della sua complessità,

ovvero i criteri sono in grado di rappresentare quanto bene il modello si adatta ai dati in proporzione al numero di parametri usato nel modello.
Nella scelta del modello da utilizzare, dunque, prediligiamo quelli che hanno valori bassi di AIC e BIC e un numero di parametri minimo.
Per trovare i valori migliori di p e di q per la serie temporale:

- generiamo i modelli $ARIMA$ con $p = [1, 2, 3]$, $q = [1, 2, 3]$ e $d = 0$;
- calcoliamo di ogni modelli i valori AIC e BIC;
- scegliamo il modello che minimizza i valori AIC e BIC con il minimo numero di parametri possibili.

Per aiutarci nella scelta riportiamo, nella tabella 3-4, l'elenco dei modelli istruiti e dei relativi valori AIC e BIC e creiamo il diagramma 3-5 che ci permette di capire quale modello (asse delle ascisse) ha valore AIC e BIC minimo (valori riportati sull'asse delle ordinate).

model_nbr	p	d	q	AIC	BIC
0	0	0	0	775.18	783.08
1	0	0	1	478.27	490.12
2	0	0	2	331.95	347.76
3	0	0	3	237.62	257.37
4	1	0	0	42.78	54.63
5	1	0	1	-2.95	12.85
6	1	0	2	-7.89	11.87
7	1	0	3	-12.13	11.57
8	2	0	0	-4.16	11.64
9	2	0	1	-4.69	15.06
10	2	0	2	-10.83	12.87
11	2	0	3	-10.18	17.48
12	3	0	0	-6.23	13.53
13	3	0	1	-4.87	18.84
14	3	0	2	-11.02	16.64
15	3	0	3	-9.55	22.05

Figura 3-4: Tabella AIC-BIC

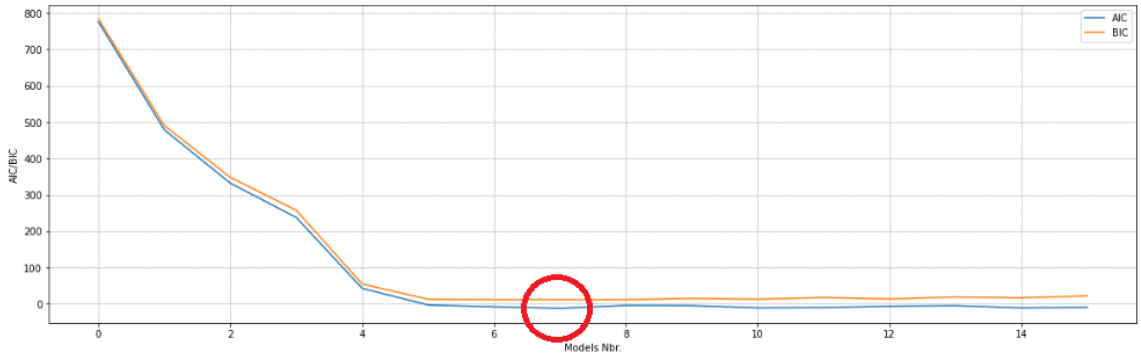

Figura 3-5: Diagramma AIC-BIC

Deduciamo dalla tabella 3-4 e dal diagramma 3-5 che il miglior modello $ARIMA$ è quello con $p = 1$ e $q = 3$.

3.4.4 Predizione

Istruiamo un modello $ARIMA(1, 0, 3)$ sul dataset di train della serie temporale in oggetto. Prediciamo i punti del dataset di test giorno per giorno. Il risultato è quello di figura 3-6 e presenta $MSE = 0.0166662$, $RMSE = 0.129098$ e $MAE = 0.0982684$

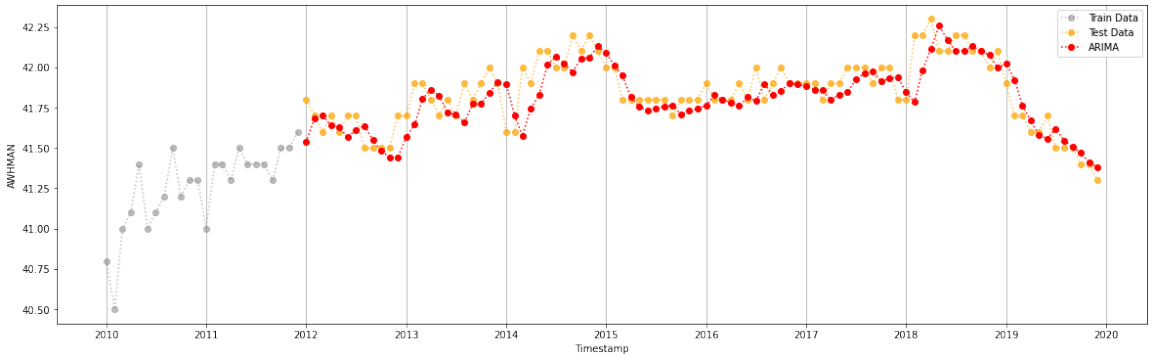

Figura 3-6: Predizione ARIMA(1,0,3)

Capitolo 4

Recurrent Neural Network

In questo Capitolo descriviamo l'architettura di una rete neurale ricorrente (RNN). Infine, vediamo l'applicazione di una rete RNN alla serie storica in oggetto e i risultati ottenuti.

4.1 Reti Neurali Ricorrenti - Simple RNN

Le reti neurali ricorrenti si differenziano dalle reti feed forward in quanto ammettono una dipendenza temporale dei dati. Ciò le rende particolarmente utili per la predizione di serie storiche dove, il dato corrente, è relazionato al dato passato.
Se nelle reti feed forward il flusso di dati procede dai livelli di input fino a quelli di output in maniera lineare, ovvero l'output di un livello diventa l'input del livello successivo, nelle reti RNN, invece, l'output torna ad essere l'input dello stesso livello. In questo modo viene inserita una dimensione temporale nella rete in quanto l'output emesso al tempo t dalla rete torna come ingresso, alla stessa rete, al tempo $t + 1$.
Una RNN, dunque, e' simile alle reti classiche, eccezion fatta per il collegamento che riporta il dato uscente dal nodo $h^{(t)}$ come input a sè stesso (vedi figura 4-1).

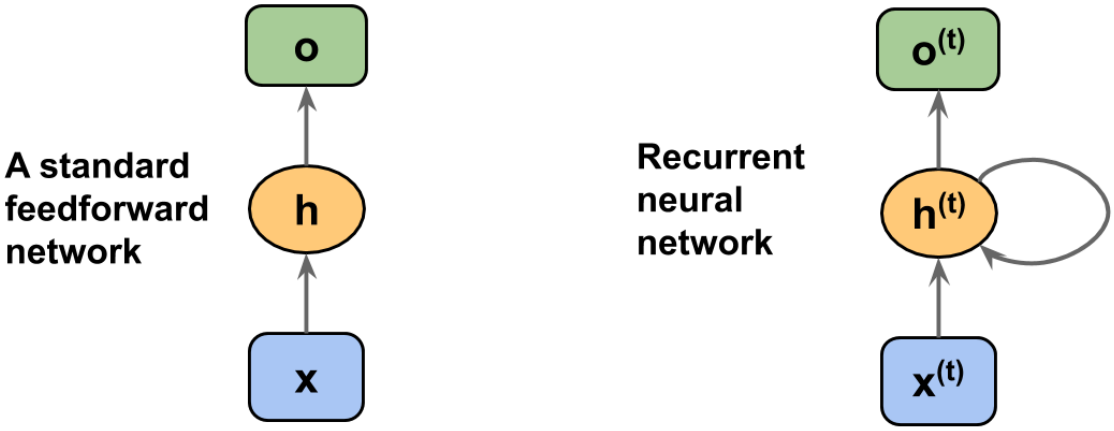

Figura 4-1: Rete Feed forward e Rete Neurale Ricorrente

Nell'esempio di figura 4-1: la rete $h^{(t)}$, ad ogni istante temporale t, riceve in ingresso l'input di $x^{(t)}$ e $h^{(t-1)}$.

4.1.1 Unfolding della rete

La rappresentazione della RNN come da immagine 4-1 viene sostituita da una versione unfolded (vedi immagine 4-2). Grazie all' unfolding rendiamo esplicite le dipendenze temporali tra l'hidden layer al tempo t e quello al tempo $t-1$; ciò ci permette di applicare più facilmente l'algoritmo di backpropagation della funzione di loss e dunque di aggiornare i pesi delle matrici associate agli hidden layer.

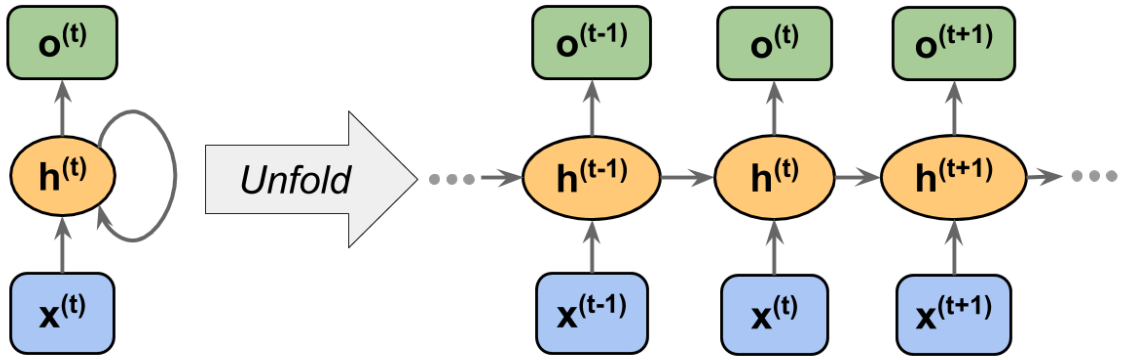

Figura 4-2: Versione Unfolded di una Rete Neurale Ricorrente

Più in dettaglio, dal confronto tra l'output predetto e quello reale calcoliamo la funzione loss e, essendo tale funzione differenziabile, il relativo gradiente. La rappresentazione unfolded ci permette di distribuire il gradiente mediante backpropagation con le modalità che abbiamo già visto per le altre reti neurali.
Dalla figura 4-2 possiamo osservare che l'output dell'hidden layer al tempo $t+1$ influenza i pesi dei livelli ai tempi precedenti (arrivando fino all'hidden layer di $t-1$ e aggiornando i pesi dei layer incontrati). Questa influenza è però di medio termine in quanto gli hidden layer più lontani verranno sempre meno influenzati (fenomeno del gradient vanishing 4.1.3).
Dalla figura 4-2 notiamo anche che i gradienti, che contribuiscono al tempo t a modificare i pesi dell'hidden layer $h^{(t)}$, sono due: quello del gradiente calcolato al passo $t+1$ e quello calcolato al passo t; quest'ultimo in maniera preponderante, il primo meno.

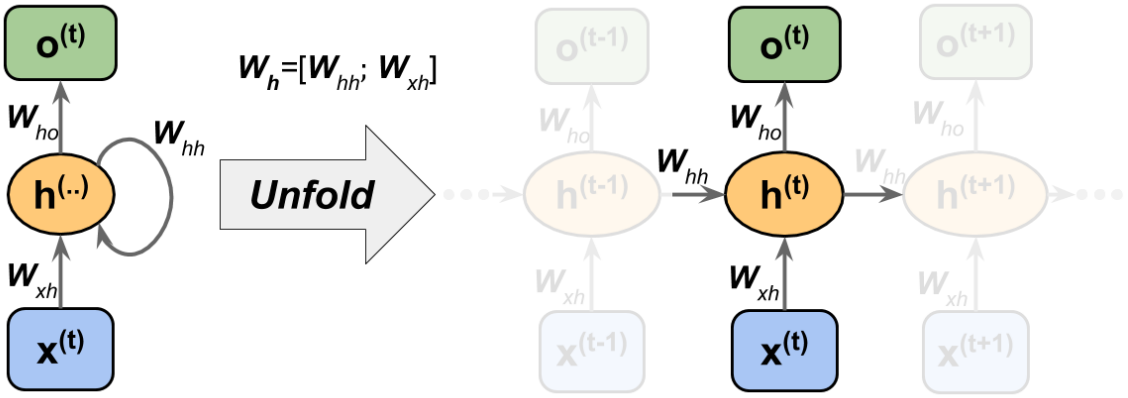

Figura 4-3: Versione Unfolded di una Rete Neurale Ricorrente

L'unfolding della RNN ci aiuta infine anche ad identificare le tre matrici dei pesi associate a ciascun hidden layer che devono essere apprese durante la fase di addestramento della rete (vedi figura 4-3):

- la matrice dei pesi tra il livello hidden e il layer di output W_{ho}
- la matrice dei pesi associata al loop attorno al layer ricorrente W_{hh}

- la matrice dei pesi che caratterizzano la connessione tra il layer di input e quello ricorrente: W_{xh}

4.1.2 Training RNNs

Per addestrare una RNN sfruttiamo la versione unfolding della rete e applichiamo la tecnica di backpropagation. Più in dettaglio, ad ogni predizione eseguita dell'output della RNN, confrontiamo il valore predetto con quello reale, calcoliamo la funzione di loss e il relativo gradiente e su tale eseguiamo tramite backpropagation l'aggiornamento dei pesi delle matrici associate agli hidden layer che precedono il layer di output.

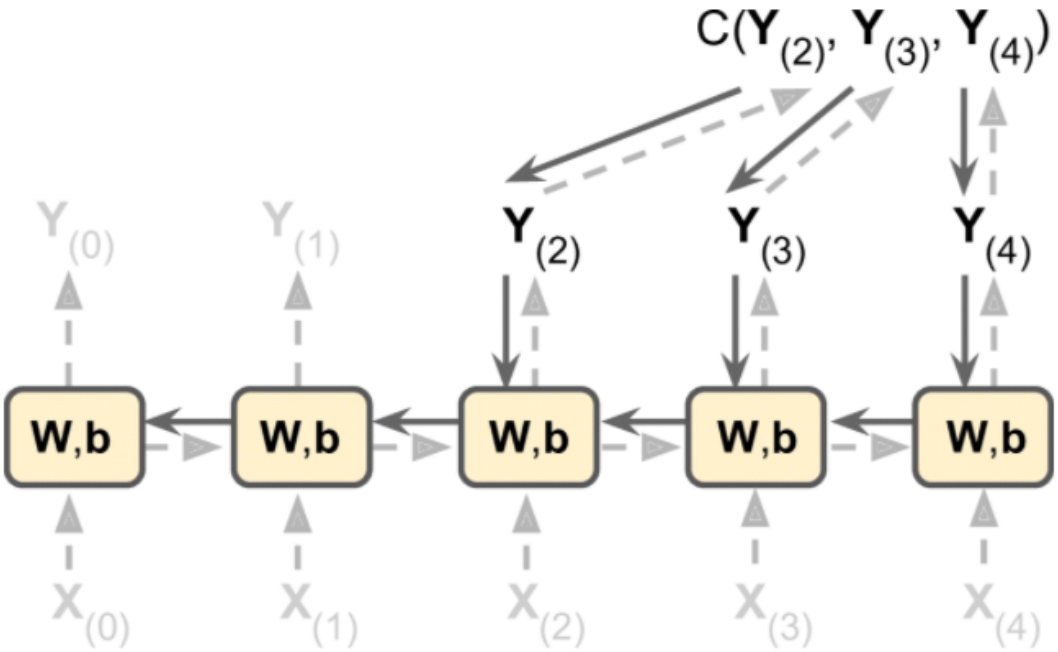

Figura 4-4: Versione Unfolded di una Rete Neurale Ricorrente

4.1.3 Vanishing/Exploding Gradient

Durante la fase di addestramento delle reti neurali, abbiamo visto che viene utilizzato il metodo della backpropagation del valore del gradiente per aggiornare i pesi degli hidden layer.

Purtroppo esiste un fenomeno di attenuazione e di moltiplicazione del valore del gradiente man mano ci allontaniamo dall'hidden layer di livello t. Tale fenomeno è a causa delle funzioni di attivazione che utilizziamo all'interno dei livelli.
Se utilizziamo funzioni di attivazione del tipo sigmoidale o tangente iperbolica, il valore del gradiente diminuisce velocemente lungo la catena dei layer (vanishing gradient); mentre, se utilizziamo funzioni lineari come la ReLU o la lineare, il valore del gradiente può crescere enormemente (exploding gradient).
Per limitare gli effetti di vanishing e exploding gradient utilizziamo gli hidden layer LSTM (Long-Short Term Memory) o i GRU (Gatered Recurrent Unit). Per motivi di spazio, non approfondiamo in questa tesi lo studio di questi tipi di celle e rimandiamo alle seguenti risorse online eventuali approfondimenti [2] e [3].

4.2 Rete Neurale Ricorrente e Risultati

In quest'ultima sezione descriviamo la rete neurale ricorrente costruita e i risultati ottenuti sulla serie storica oggetto della tesi.

4.2.1 RNN

La rete neurale costruita è così composta:

- ***Input Layer*** - il layer di input è rappresentato da un vettore di 10 elementi di dimensione 1, ovvero, ipotizzando di essere al tempo T, dai dieci elementi della serie storica $[T-9, T-8, ..., T-1, T]$.
- ***LSTM*** - il layer hidden è composto da 24 unità LSTM;
- ***Output Layer*** - il layer di output è un vettore di dimensione 1 contenente il valore della serie storica a $T+1$.

Riassumiamo la rete neurale costruita con il seguente sommario:

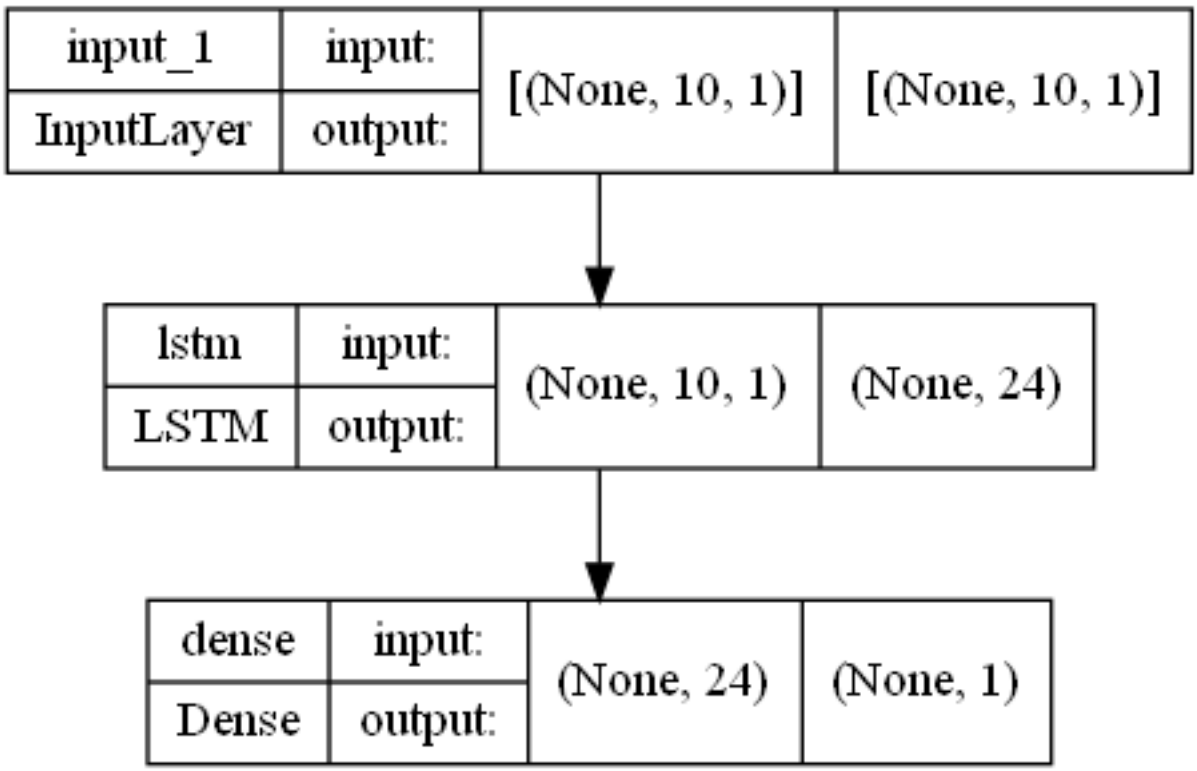

Figura 4-5: Sommario della Rete Neurale Ricorrente

Terminiamo la configurazione della rete scegliendo, come funzione di loss, la funzione mean_squared_error, e come funzione di ottimizzazione, la ADAM.

4.2.2 Dataset di Train/Validation e Test

Il dataset originario è stato suddiviso seguendo questo criterio.
Innanzitutto, abbiamo diviso il dataset in due sotto insiemi: train e test, dopodichè il train è stato suddiviso nuovamente in train e validation.
Quindi abbiamo usato: il train per istruire la rete neurale e il validation per validarla.
Infine, abbiamo esaminato le prestazioni del modello di rete neurale, utilizzando la parte di test e valutandone le metriche MSE e RMSE.
Notiamo che, affinchè le funzioni di lost, calcolate sul train e sul validation dataset, convergessero, abbiamo dovuto eseguire 250 epoche per istruire il modello di rete.

Di seguito, il grafico riguardante i valori delle funzioni di loss del train e del validation dataset calcolate durante la costruzione del modello.

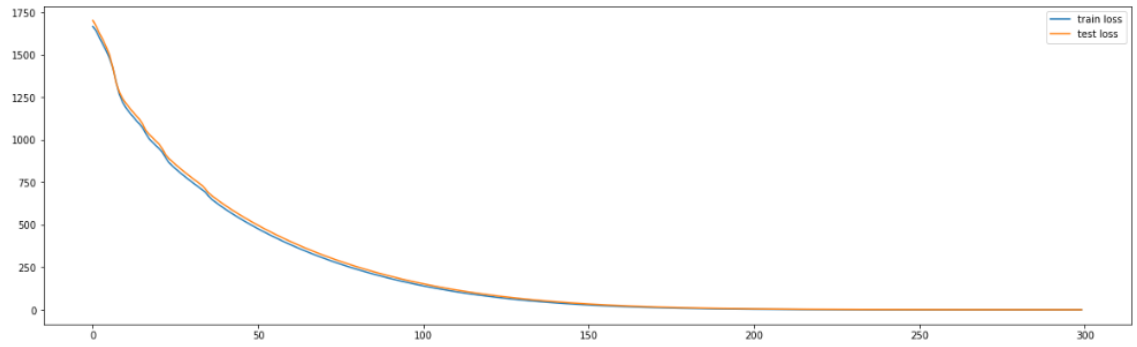

Figura 4-6: Convergenza della funzione di Loss

4.2.3 Risultati

I risultati ottenuti sul dataset di test stimando la rete dopo ogni predizione, ovvero con tecnica "One step forecast con re-estimation", sono i seguenti:

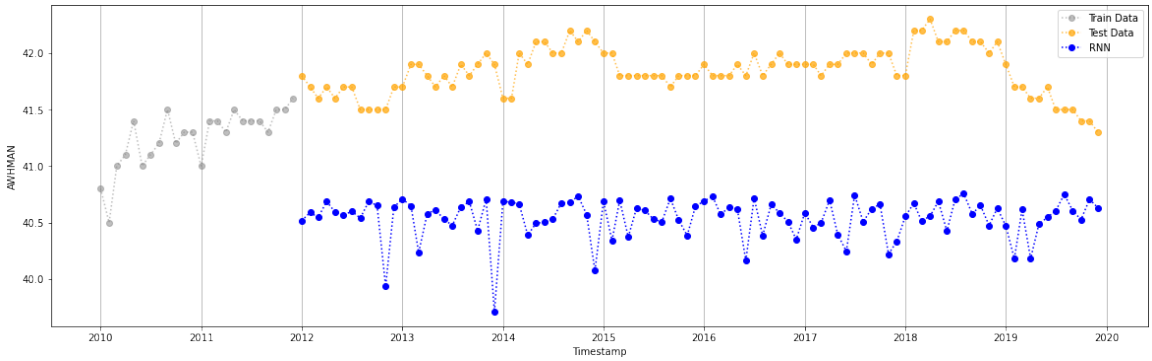

Figura 4-7: Predizione con Rete Neurale Ricorrente con re-estimation

- MSE: 1.9147
- RMSE: 1.38373
- MAE: 1.34882

Capitolo 5

Prophet

In questo capitolo descriviamo la libreria di Facebook Prophet.
Infine, vediamo l'applicazione di tale libreria alla serie storica in oggetto e i risultati ottenuti.

5.1 Introduzione

Prophet è una libreria sviluppata da Facebook che si basa su un metodo general additive, ovvero quello di rappresentare la serie storia come somma di quattro componenti.

$$y(t) = trend(t) + seas(t) + holiday(t) + \varepsilon(t) \tag{5.1}$$

dove:

- $trend(t)$ - rappresenta la componente di trend; ciò che non può essere interpretato come cambiamento periodico nel tempo;

- $seas(t)$ - descrive le possibili stagionalità presenti nella serie, ovvero modella i cambiamenti che si ripetono ad intervalli regolati di tempo; è possibile avere più di una stagionalità nella stessa serie;

- $holiday\,(t)$ - rappresenta la componente degli eventuali periodi di vacanza che possono essere configurati a priori; componente che modella eventi irregolari che modificano temporaneamente la serie temporale;

- $\varepsilon\,(t)$ - rappresenta il termine dei residui, ovvero tutto ciò che il modello non è in grado di catturare della serie temporale; $\varepsilon\,(t)$ viene rappresentata come una distribuzione normale

Alcuni vantaggi della libreria Prophet, sono:

- non dobbiamo tener conto di eventuali dati mancanti dopo aver eliminato gli outlier;

- possiamo fare assunzioni sui trend e impostare diversi livelli di stagionalità a secondo del fenomeno che stiamo studiando;

- possiamo aggiungere componenti additive esterne (per esempio dei regressori) che possono aiutare nel prevedere la serie storica;

- moduliamo l'importanza dei fattori in modo indipendente.

Vediamo in dettaglio le diverse componenti della libreria Prophet.

5.2 Trend

Il funzionamento di Prophet pone molta attenzione sulla componente descrivente il trend. In particolare, Prophet è in grado di descrivere il trend come un "Modello di staturazione non lineare", o un "Trend lineare con change points".

5.2.1 Modelli di saturazione non lineare

Possiamo utilizzare la libreria Prophet per descrivere dei modelli di saturazione non lineare.
Tali modelli descrivono una popolazione in crescita che continuerà ad aumentare fino

a un certo livello di saturazione (dove tale livello può anche aumentare nel tempo). Per esempio, per i mercati finanziari, possiamo prevedere in quale momento il mercato verrà saturato conoscendone la capacità massima.

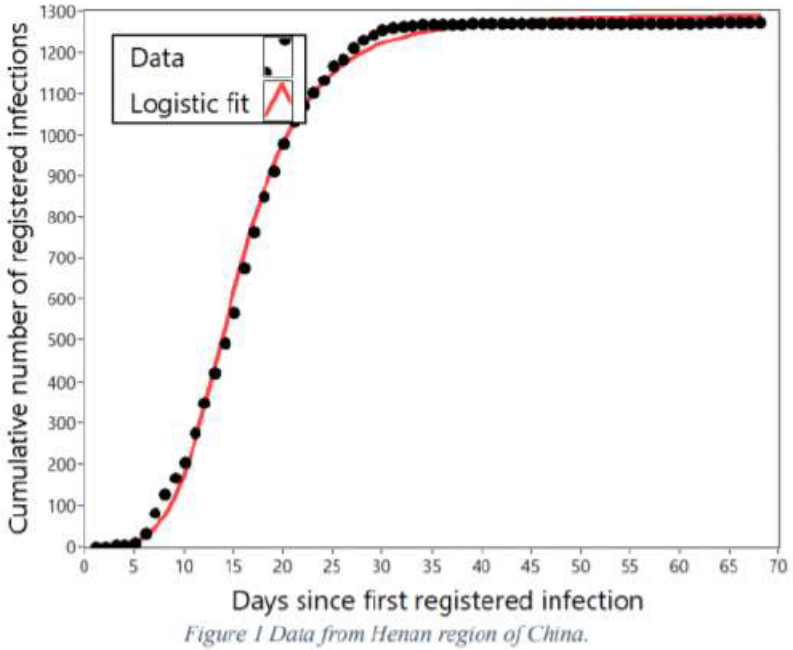

Figure 1 Data from Henan region of China.

Figura 5-1: Esempio di Mercato a saturazione

Questo tipo di trend può essere rappresentato con questa formula:

$$g(t) = \frac{C}{1 + \exp(-k(t - m))} \tag{5.2}$$

Dove C è la carrying capacity (il limite alla crescita della serie temporale), k è il tasso di crescita e m un parametro di offset. C e k possono non essere costanti ma essere a loro volta delle variabili dipendenti dal tempo.

5.2.2 Trend lineari con change points

Con Prophet possiamo anche modellare serie temporali che presentano, in determinati periodi, andamenti crescenti o descrescenti. I punti in cui gli andamenti della serie si invertono possono essere inseriti manualmente o inferiti in maniera automa-

tica attraverso un'analisi Bayesiana. Questa analisi permetterà anche di prevedere l'andamento futuro dei cambiamenti.

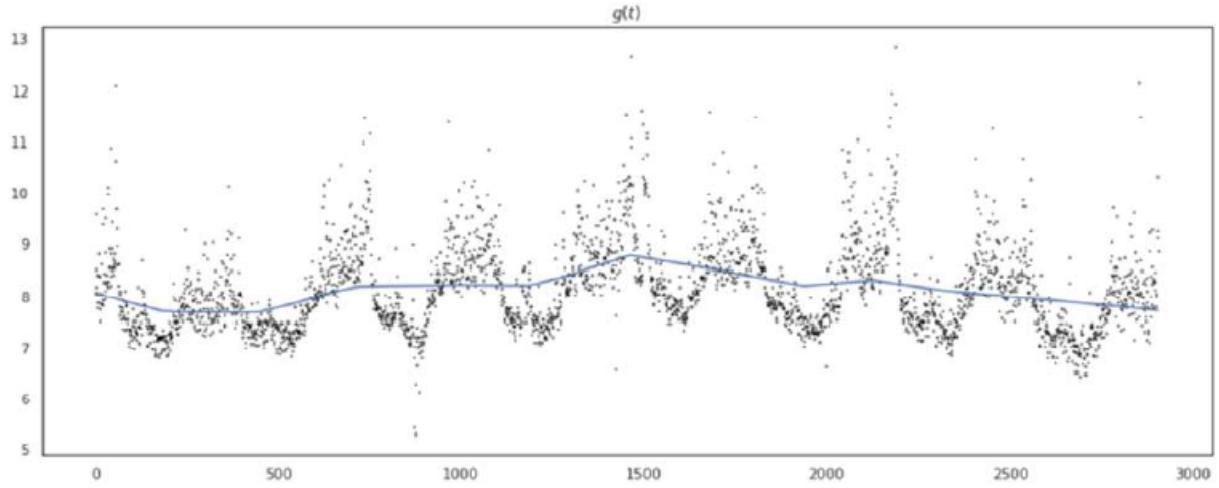

Figura 5-2: Esempio di Serie Temporale con Change Points

5.3 Stagionalità

Un'altra componente importante su cui la libreria Prophet pone molta attenzione è quella della stagionalità. Prophet prevede la presenza di molteplici componenti stagionali: annuali, settimanali, giornaliere e sub-giornaliere. Per modellare queste stagionalità, Prophet, si basa sul teorema di Fourier che permette di rappresentare un segnale periodico come somma di altri segnali periodici.

$$seas\,(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n + \sin\left(\frac{2\pi nt}{P}\right)\right) \tag{5.3}$$

Possiamo intuire che le serie storiche, di cui la libreria Prophet è in grado di catturare al meglio la componente di stagionalità, sono quelle in grado di essere descritte come somme di componenti seno e coseno.

Osserviamo inoltre che il parametro N è un "filtro" per le stagionalità: valori alti di N portano a stimare le variazioni di stagionalità più veloci ma rischiano di costruire modelli in overfitting, mentre valori di N bassi permettono di catturare stagionalità

più lente e andamenti più macroscopici della stagionalità e, di conseguenza, evitare l'overfitting.

5.4 Holiday

Con holiday intendiamo eventi eccezionali nella serie temporale che "rompono" la stagionalità. Questi eventi non possono essere rilevati automaticamente dal modello, in quanto per definizione sono "imprevedibili"; devono quindi essere definiti manualmente.

5.5 Setup di Prophet e Risultati

In quest'ultima sezione descriviamo il setup della libreria Prophet e i risultati ottenuti sulla serie storica, oggetto della tesi.

5.5.1 Setup di Prophet

Durante la creazione del modello abbiamo esplicitamente: ricercato un trend lineare con change points; limitato il numero di tali punti a 20 e la ricerca di essi nell'80% iniziale del dataset di train; infine, abbiamo disabilitato la ricerca di stagionalità settimanali e giornaliere, in quanto i dati in uso sono mensili e quindi dotati di una granularità superiore.
Il modello risultante è il seguente:

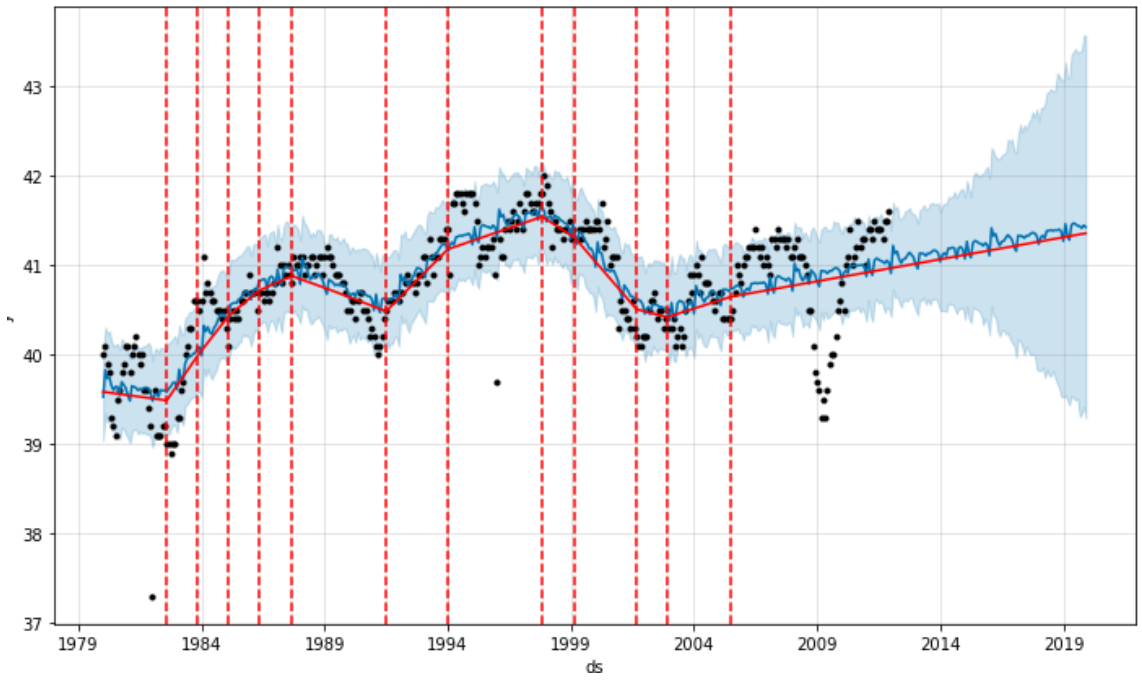

Figura 5-3: Modello Prophet

Dotato delle componenti:

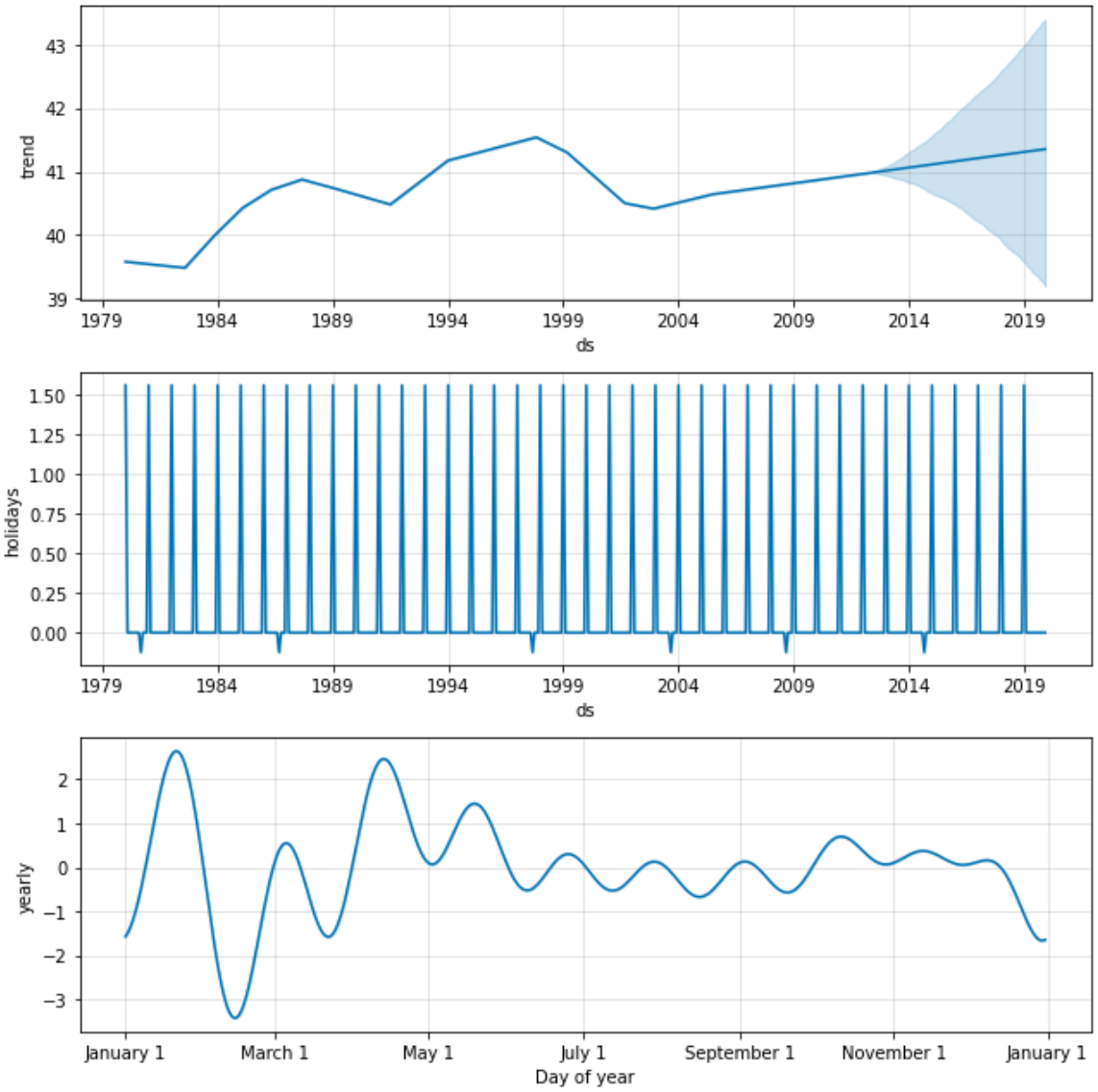

Figura 5-4: Componenti del Modello Prophet

5.5.2 Risultati

Per predirre i valori appartenenti al dataset di test abbiamo utilizzato la tecnica "One step forecast con re-estimation". I risultati sono i seguenti:

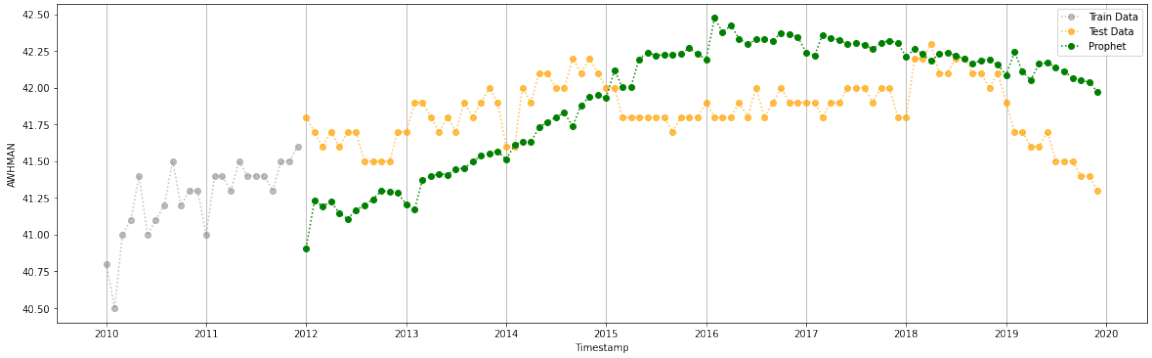

Figura 5-5: Predizione con Prophet con re-estimation

- MSE: 0.166088
- RMSE: 0.407539
- MAE: 0.365322

Capitolo 6

Conclusioni

In questa tesi, abbiamo preso in considerazione la serie storica rappresentante la media delle ore settimanali lavorative nel settore Manufacturing, da Gennaio 1980 a Dicembre 2019.
Abbiamo analizzato graficamente la serie storica e abbiamo applicato ad esse i seguenti modelli predittivi: ARIMA, RNN e Prophet. Per ciascun modello, abbiamo, infine, calcolato le metriche di MSE, RMSE e MAE. Il miglior modello si è rivelato essere ARIMA.

Modello	**MSE**	**RMSE**	**MAE**
ARIMA	0.0166662	0.129098	0.0982684
RNN	1.9147	1.38373	1.34882
Prophet	0.166088	0.407539	0.365322

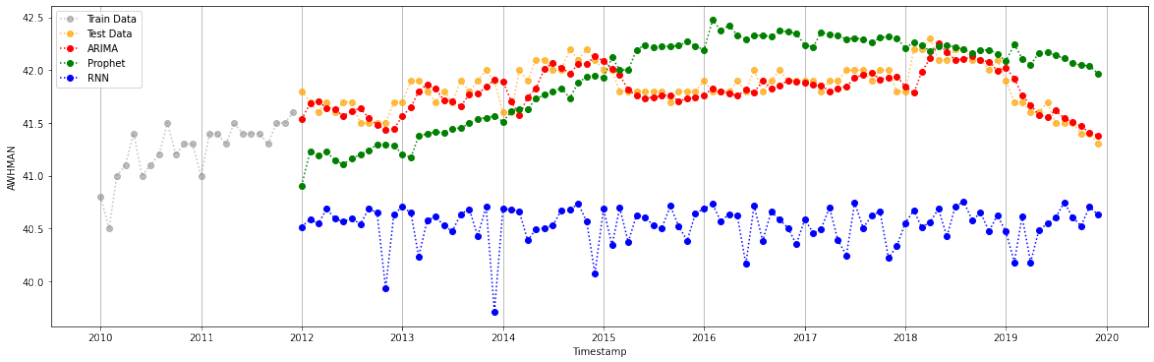

Figura 6-1: Predizioni con re-estimation

Osservazioni: ARIMA è un modello molto potente che riesce a generalizzare bene una serie temporale, se correttamente configurato. Quindi, malgrado richieda tempo ed attenzione nella ricerca dei parametri p, d e q, al termine fornisce un modello molto valido. Un aiuto nel compito di ricerca dei parametri può arrivare dalla libreria AutoARIMA, che semplifica il processo di ricerca mediante una ricerca esaustiva degli stessi.

Prophet è progettato specificamente per la previsione di serie temporali, per cui, a priori, siamo in grado di dare indicazioni su determinate caratteristiche (trend, stagionalità, vacanze, eventi eccezionali). Il vantaggio di Prophet è che richiede una minore ottimizzazione dei parametri di funzionamento, poiché è in grado anche di rilevarli automaticamente.

Le reti neurali ricorrenti basate su LSTM sono probabilmente l'approccio più potente all'apprendimento da dati sequenziali e le serie temporali sono solo un caso speciale. Il potenziale dei modelli basati su LSTM è pienamente rivelato quando si apprende da enormi set di dati, in cui possiamo rilevare modelli complessi. A differenza di ARIMA o Prophet, le RNN, non si basano su ipotesi specifiche sui dati come la stazionarietà per ARIMA o trend e stagionalità per Prophet. Uno svantaggio delle RNN basate su LSTM è che sono difficili da interpretare e ottenere intuizioni sul loro comportamento. Inoltre, per ottenere buoni risultati è necessario un set di dati di numero elevato e un'attenta messa a punto dei parametri, quali il numero di unità di input e di layer hidden.

Bibliografia

[1] Version: 0.14.0 Library: statsmodels. *Function: seasonal_decompose*. https://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.seasonal_decompose.html/. Statsmodels Documentation.

[2] Christopher Olah. *Understanding LSTM Networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Blog.

[3] Simeon Kostadinov. *Understanding GRU Networks*. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be. Blog.