Relazione 3
Introduzione alla Statistica Computazionale

Matteo Scardala
m.scardala@studenti.unipi.it

Dicembre 2024

1 Introduzione

Supponiamo che una multinazionale energetica debba pianificare la quantità di risorse necessarie per il 2025. Per far ciò è necessario stimare la domanda nei vari paesi in cui opera, ad esempio la Grecia. Lo scopo di questo progetto è studiare una serie storica del consumo energetico in Grecia al fine di fare questa previsione.
A questo proposito utilizzo dati mensili relativi agli ultimi 16 anni analizzati con un programma python riportato in appendice.

2 I dati

I dati utilizzati per l'analisi sono stati forniti da Eurostat, l'ufficio statistico dell'UE. I dati presi in esame riportano il consumo mensile di energia in Grecia, espresso in GWh, a partire dal primo gennaio 2008 fino al primo ottobre 2024. La tabella si può trovare al seguente link.
La tabella ha diversi attributi, ma ho costruito la serie temporale scegliendo le date come colonna di indici della serie e `OBS_VALUE` come i valori della serie.

3 Analisi della serie

In questa sezione verrà analizzata la serie, cercando di cogliere (se presenti) le componenti di trend, stagionalità e rumore. Per prima cosa è utile fornire una panoramica dei dati, illustrando il grafico delle serie e della funzione di autocorrelazione.

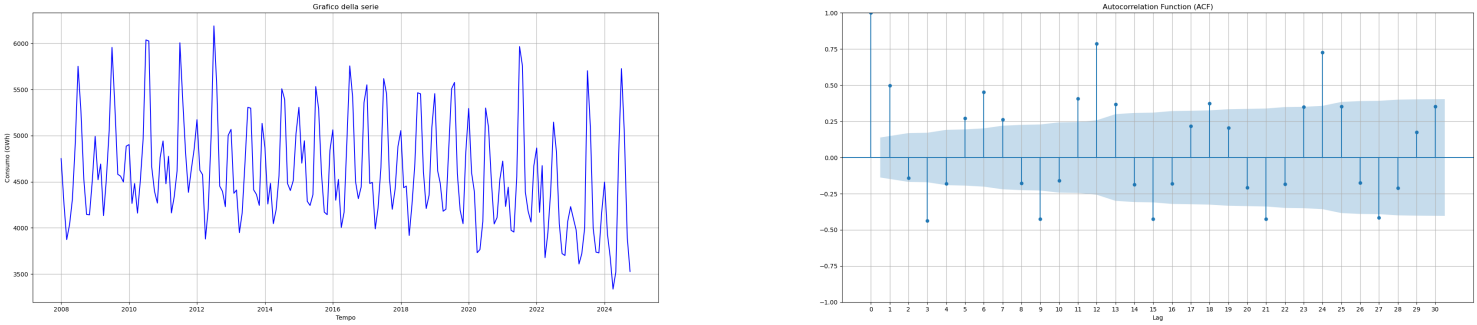


Figura 1

Dal primo grafico notiamo una probabile periodicità, un trend che inizialmente sembra abbastanza costante per poi scendere leggermente verso gli anni più recenti. Per quanto riguarda le ampiezze dei "picchi", si nota un periodo centrale in cui sono meno ampie: per questo motivo farò in seguito un' analisi della stazionarietà della periodicità.
Nel secondo grafico la banda azzurra mostra la regione in cui la funzione è statisticamente nulla (al 95%). Sono evidenti dei picchi ai lag 12 e a 24, cioè la serie ha molto probabilmente stagionalità di periodo 12 (un anno).

Figura 2

Un altro indizio a riguardo sono i grafici precedenti (Figura 2), che mostrano le differenze consecutive della serie e la relativa ACF. Notiamo che anche dal grafico dell'ACF delle differenze ci sono dei picchi a 12 e 24, dunque è maggiormente avvalorata l'ipotesi di stagionalità con periodo 12.
Passiamo alla decomposizione della serie. Nei seguenti grafici vengono riportati i risultati per quanto riguarda la decomposizione additiva e moltiplicativa, scegliendo come periodo 12, per quanto discusso in precedenza.

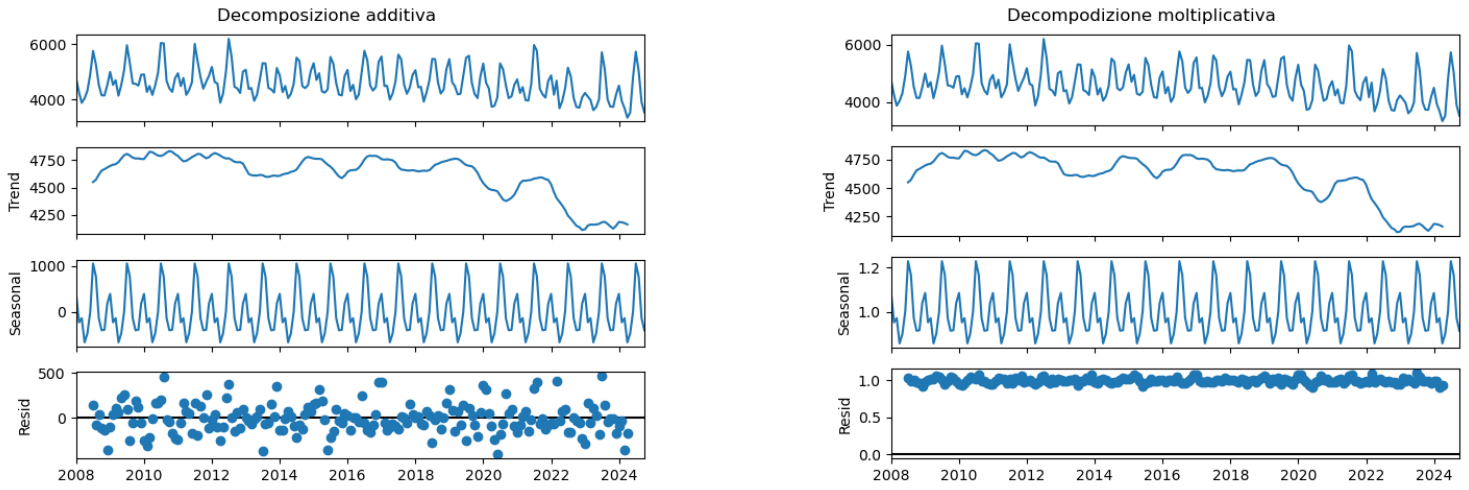


Figura 3

A questo punto si può concludere che la serie ha un carattere stagionale, dato che nella decomposizione additiva la parte stagionale ha un'ampiezza paragonabile ai valori della serie, mentre nella decomposizione additiva oscilla fino a 1.2. Il trend sembra inizialmente oscillare tra 4500 e 4750 per poi iniziare un declino attorno al 2020. Per capire quale modello descrive meglio la serie è necessario analizzare la parte di rumore, ma questo verrà fatto alla fine della sezione. Infatti vorrei prima valutare la stazionarietà della componente stagionale: preliminarmente fornisco due grafici.

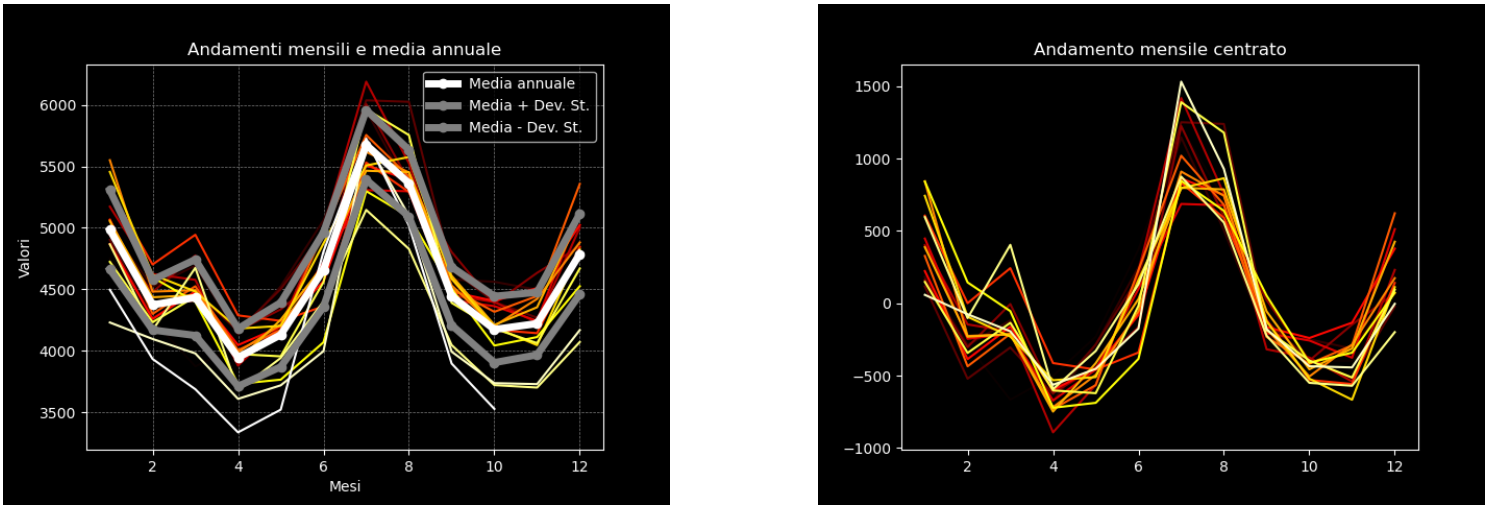


Figura 4

Il primo riporta sulle ascisse i mesi dell'anno e i valori della serie sulle ordinate. Ogni anno viene rappresentato con un colore diverso: dal rosso scuro al bianco via via che gli anni diventano più recenti. Inoltre vengono riportare le medie di ogni mese e relative deviazioni standard (per novembre e dicembre vengono ignorati i valori del 2024 dato che non fanno parte della serie). Si osserva che gli anni più recenti hanno valori sensibilmente minori: per gli ultimi 3 anni molti valori sottostanno addirittura alla banda grigia. Questo segnala un generale lento calo del trend che si accentua negli ultimi 3 anni.
Il secondo grafico mostra i valori della serie in modo simile al grafico precedente, ma per ogni anno la serie viene centrata nella sua media annuale. Dal grafico si nota che il picco estivo è meno ampio per gli anni "centrali", come abbiamo notato nel grafico preliminare. Tuttavia, tranne altri punti meno evidenti, i grafici risultano abbastanza compatti, dunque dalla semplice analisi grafica non saprei dare una risposta perentoria sulla stazionarità dei periodi: direi che c'è una stazionarietà debole in quanto la differenza nella componente stagionale è apprezzabile solo nei massimi annuali.
Dato che la stazionarietà dei periodi sembra debole, ho fatto decomposizione STL mediando su 7 unità temporali, sia per un modello additivo, che per un modello moltiplicativo: per quest'ultimo ho fatto decomposizione STL del logaritmo della serie e poi mostrato sul grafico gli esponenziali dei risultati.

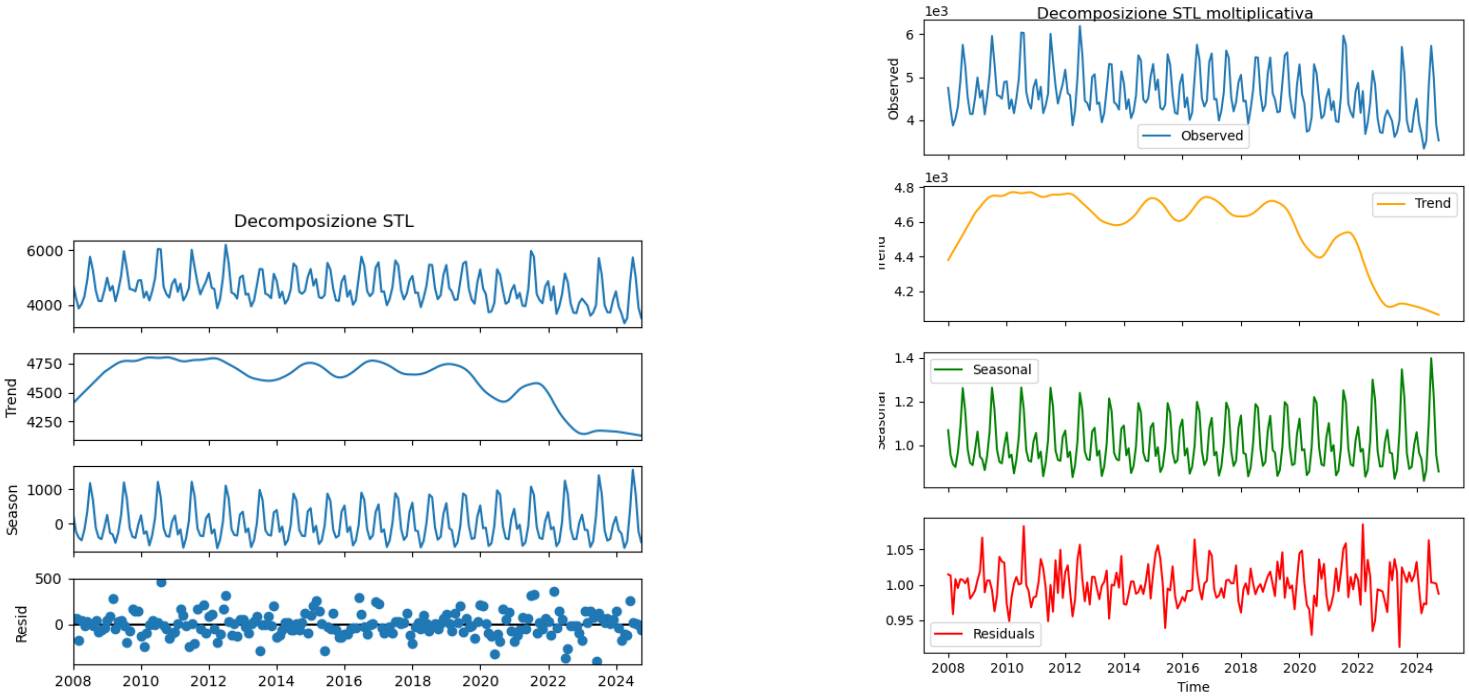


Figura 5

I risultati non sono dissimili dalla scomposizione precedene: si nota una forte componente stagionale, che in questo caso è leggermente più accentuata all'inizio e alla fine della serie.
Per concludere la sezione analizzo i residui dei modelli additivo e moltiplicativo e delle composizioni STL.

3.1 Analisi dei residui, decomposizione additiva e moltiplicativa

Nei seguenti grafici vengono confrontate le stagionalità e i residui delle due decomposizioni. Per confrontarle, le componenti moltiplicative vengono centrate in zero e moltiplicate per il trend.

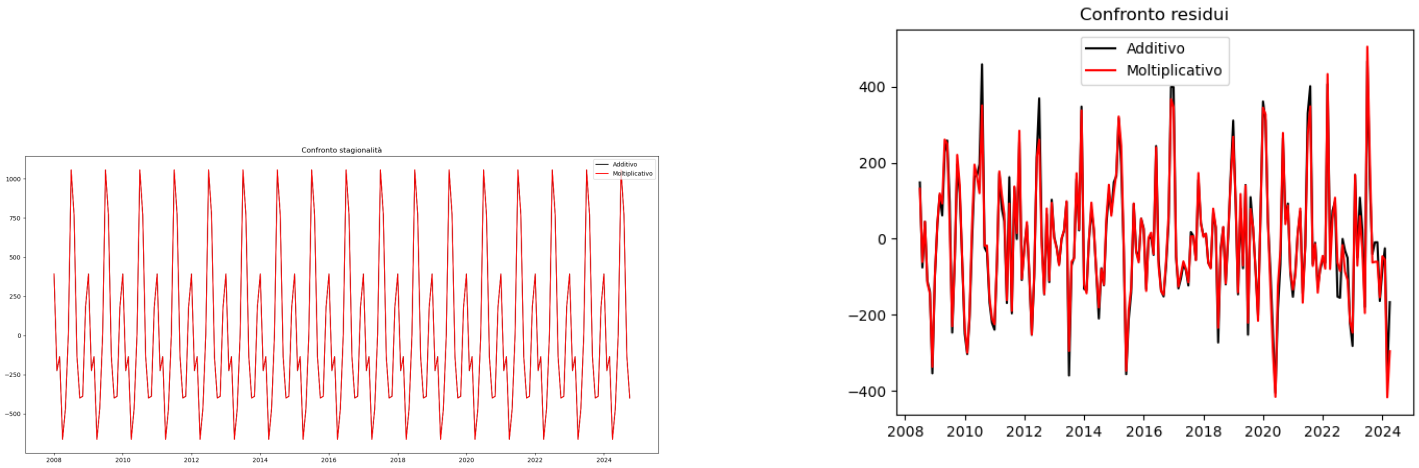


Figura 6

Notiamo che le componenti stagionali delle due decomposizioni si sovrappongono quasi perfettamente, mentre le componenti dei residui mostrano un risultato simile, con una sovrapposizione meno accentuata.
Di seguito vengono mostrati i residui dei due modelli sul grafico (Figura 7). Osservando i risultati non si nota una particolare struttura nella disposizione nel piano: un primo indizio del fatto che la decomposizione ha estratto abbastanza struttura dai dati, lasciandone poca nella parte di rumore.
Questa ipotesi è confermata dal grafico della ACF dei residui fornito in seguito (Figura 8). In entrambi casi abbiamo che dopo pochi lag l'ACF è statisticamente nulla, dunque posso affermare con un buon grado di certezza che la composizione additiva e moltiplicativa sono entrambe buone perché hanno estratto bene la struttura dai dati, in quando i residui appaino casuali e senza correlazioni significative.

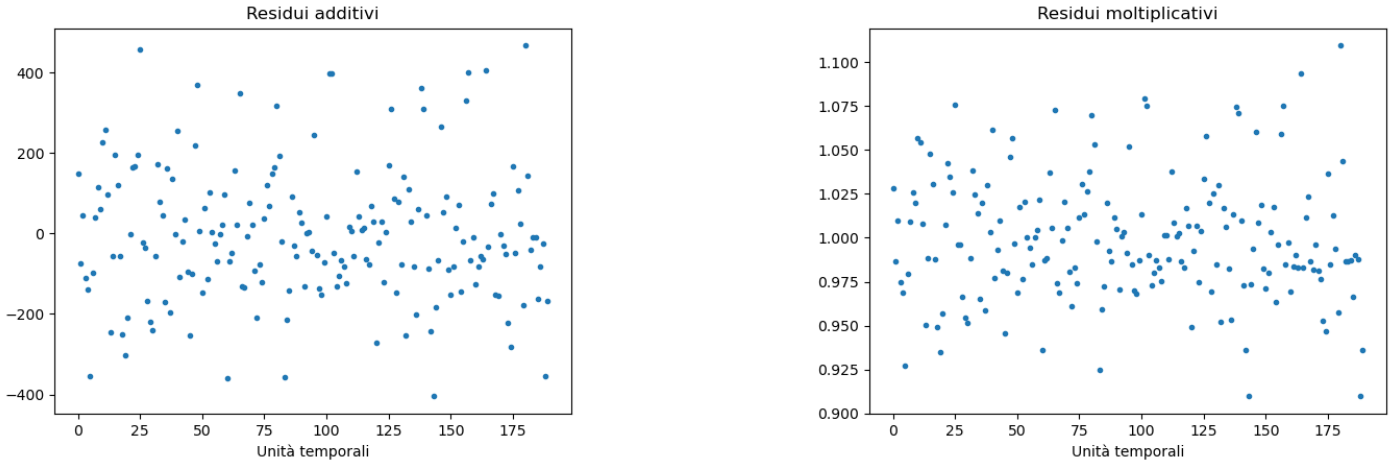


Figura 7

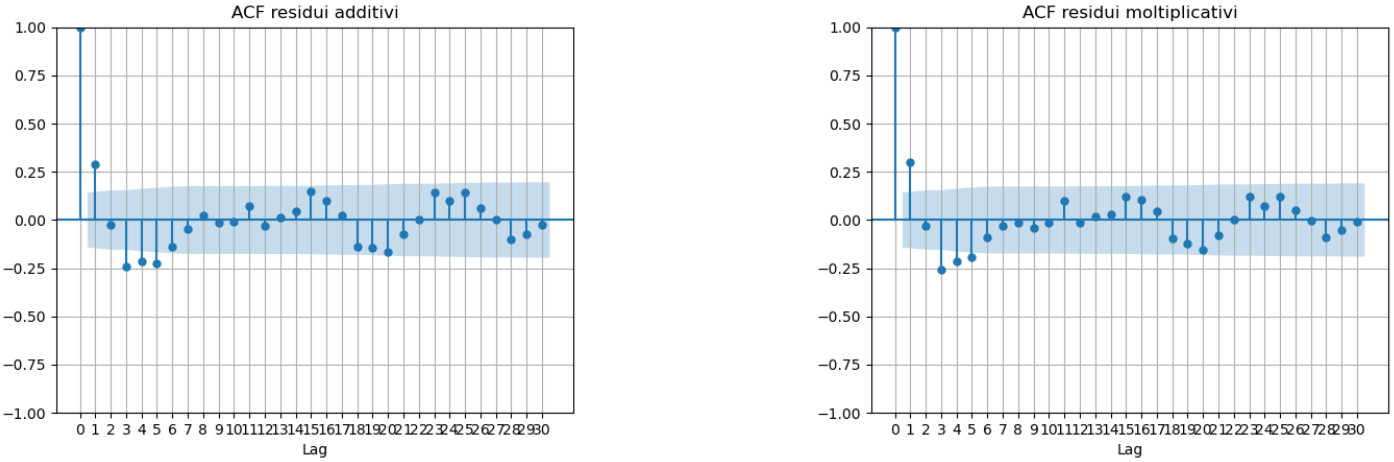


Figura 8

A questo punto sarebbe interessante analizzare l'eventuale gaussianità dei residui. Per entrambe le decomposizioni fornisco di seguito due grafici:

- Confronto tra la distribuzione dei residui e la distribuzione di una gaussiana con stessa media e varianza
- Confronto tra quantili empirici e i quantili teorici di una distribuzione gaussiana con stessa media e varianza dei residui

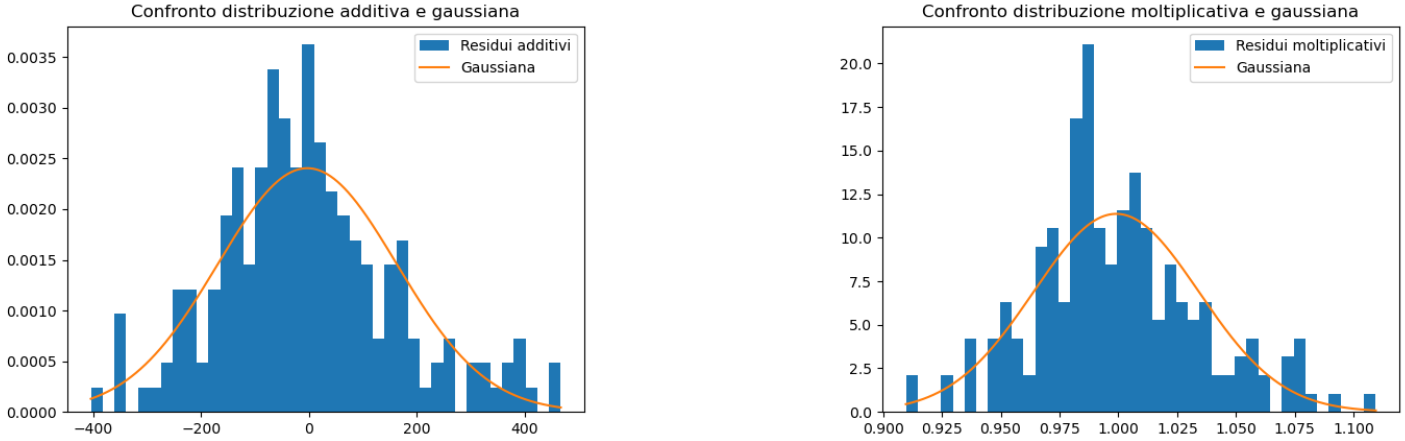


Figura 9

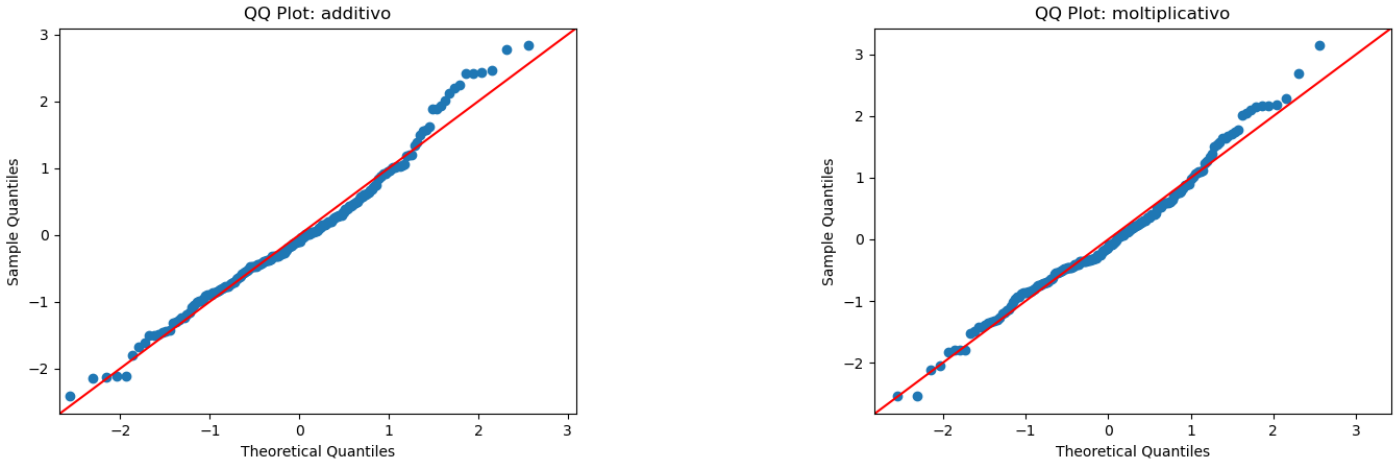


Figura 10

Osservando i grafici dei quantili tenderei a rifiutare l'ipotesi di gaussianità, soprattutto per la poca aderenza nella coda destra. Inoltre dai grafici sulle distribuzioni dei residui si nota una leggera asimmetria. Da un punto di vista quantitativo si osservano i seguenti valori

- **Modello additivo:** Skwewness: 0.4463 . Kurtosi: 0.3882. p-value di Shapiro-Wilk: 0.0061
- **Modello moltiplicativo:** Skwewness: 0.3746. Kurtosi: 0.3882. p-value di Shapiro-Wilk: 0.0099

Osservazione: in questo lavoro la kurtosi viene calcolata normalizzando i dati e sottraendo 3, dunque una distribuzione gaussiana ideale dovrebbe avere kurtosi nulla.
I risultati confermano i sospetti che derivano dai grafici, per cui rifiuto l'ipotesi di gaussianità (avevo scelto come livello di confidenza 0.05).
A questo punto per decidere quale modello sia migliore per spiegare il fenomeno, adopero criteri più quantitivi calcolando la varianza spiegata dai residui e la deviazione standard dell'ACF dei residui. Poiché i residui delle due composizioni hanno significati diversi, per fare il confronto vedo il modello moltiplicativo come un modello additivo tra i logaritmi, cioé

$$x = trend \cdot seas \cdot res \Leftrightarrow \log(x) = \log(trend) + \log(seas) + \log(res)$$

e dunque confronto la varianza dei residui e la varianza spiegata dei residui del modello additivo e del modello logaritmico.

- **Modello additivo.** Varianza spiegata: 0.08620. STD ACF: 0.24450.
- **Modello moltiplicativo.** Varianza spiegata : 0.08478. STD ACF: 0.24126.

Notiamo che i risultati quantitativi sono simili, anche se si sbilanciano leggermente a favore del modello moltiplicativo. Una possibile causa di questa sovrapposizione dei risultati può essere che il modello sia moltiplicativo e venga approssimato bene al primo ordine da quello additivo.

3.2 Analisi dei residui, decomposizione STL

La stessa analisi può essere ripetuta per la decomposizione STL, che mostra i seguenti risultati:

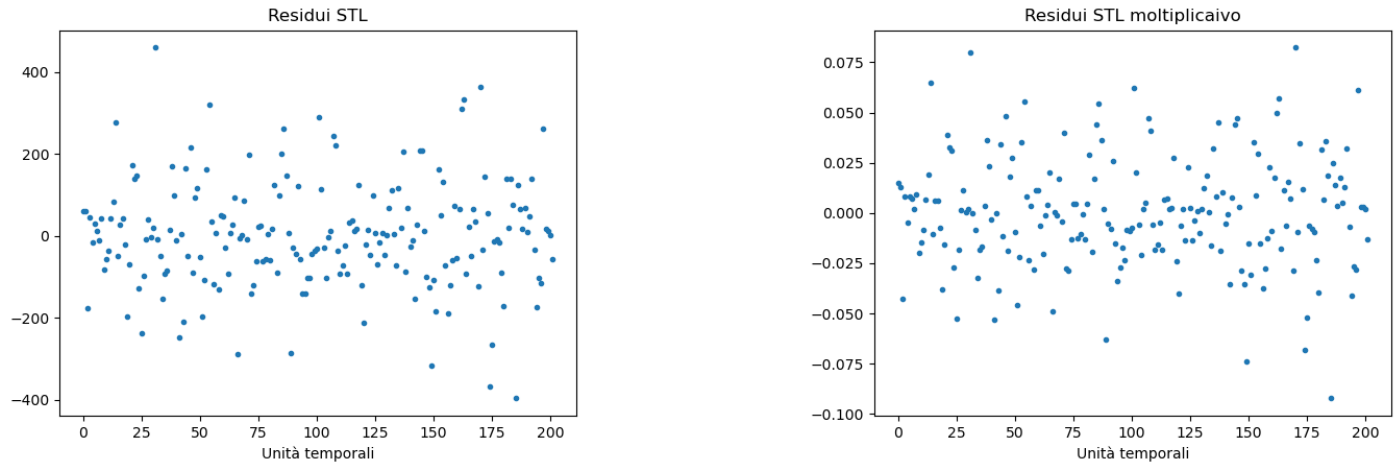


Figura 11

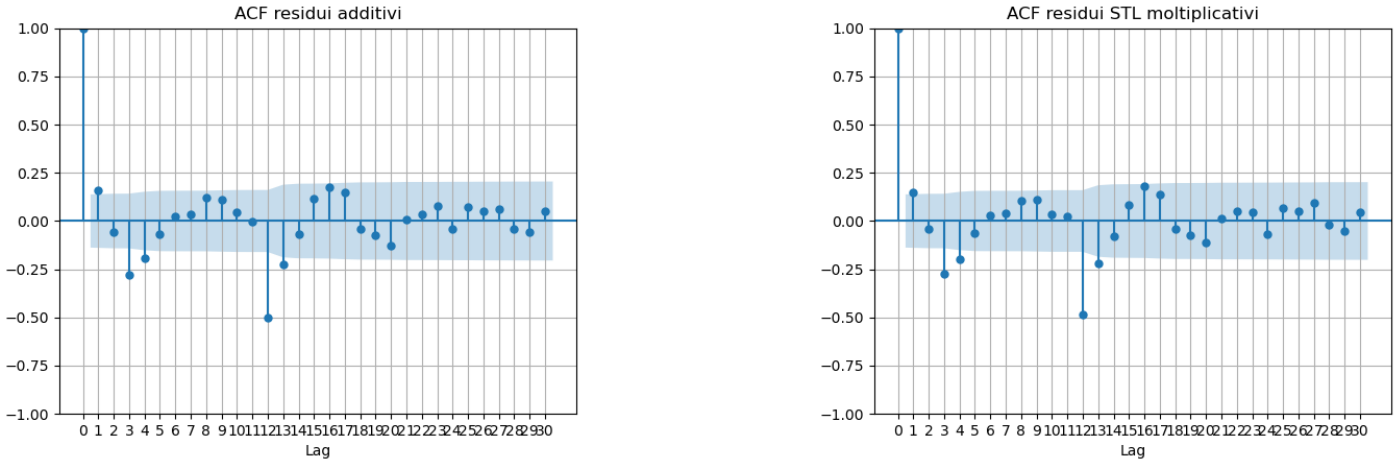


Figura 12

I residui non sembrano mostrare una particolare struttura grafica e dai grafici dell'ACF dei residui si osserva la decomposizione STL, in entrambi i casi, ha estratto molta struttura dai dati, lasciandone poca nella parte di rumore. Inoltre dal punto di vista quantitativo abbiamo:

- **Modello additivo.** Varianza spiegata: 0.0534. STD ACF: 0.2529.
- **Modello moltiplicativo.** Varianza spiegata : 0.0523. STD ACF: 0.2528.

Questi risultati, similmente a quanto visto prima, indicano che il modello è probabilmente moltiplicativo, approssimato bene al primo ordine da un modello additivo. Tuttavia, confrontando questi numeri con i risultati della decomposizione precedente, sembrerebbe che la decomposizione STL riesca a estrarre più struttura dai dati in quanto la varianza spiegata dal rumore è minore. Probabilmente si può giustificare con il fatto che la componente stagionale nella parte centrale della serie presenta delle oscillazioni di ampiezza leggermente minore. Dunque, come detto prima, la stagionalità della serie presenta un carattere di stazionarietà debole, al limite tra l'essere stazionaria e non esserlo.

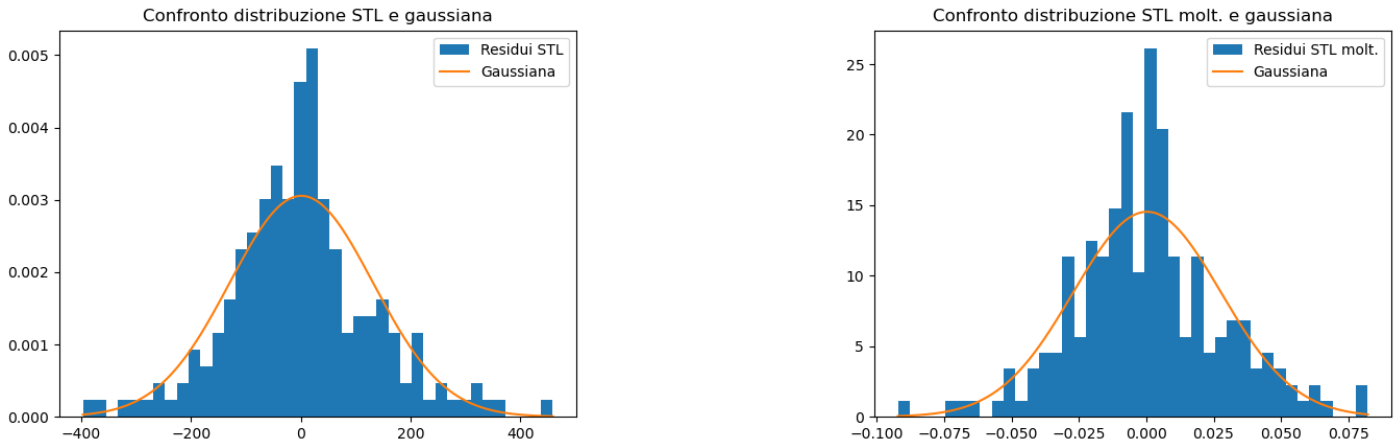
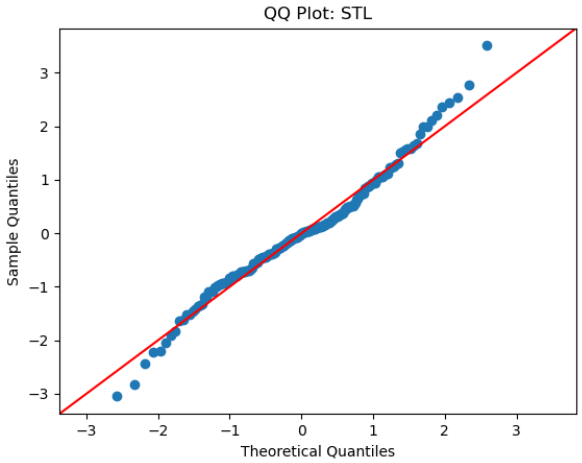


Figura 13

Per quanto riguarda la gaussianità dei residui (Figura 13), si osserva una buona aderenza nel grafico delle distribuzioni, mentre il qq-plot (Figura 14) mostra una discrepanza tra distribuzione dei residui additivi e gaussiana e una certa aderenza tra quella dei residui moltiplicativi e una gaussiana. I dati quantiativi mostrano che effetivamente è molto probabile che i residui della decomposizione STL moltiplicativa possano essere gaussiani, ipotesi non rifiutata dal test di Shapiro-Wilk a livello 95%.

- **Modello additivo STL:** Skwewness: 0.2292 . Kurtosi: 1.0716. p-value di Shapiro-Wilk: 0.009
- **Modello moltiplicativo:** Skwewness: 0.0879. Kurtosi: 0.7584. p-value di Shapiro-Wilk: 0.07

Dunque, cofrontando le varianze spiegate dai residui delle varie decomposizioni, il modo migliore per scomporre la serie sembrerebbe STL moltiplicativo, mediando su 7 lag, e questa decomposizione fornisce addirittura dei residui gaussiani.



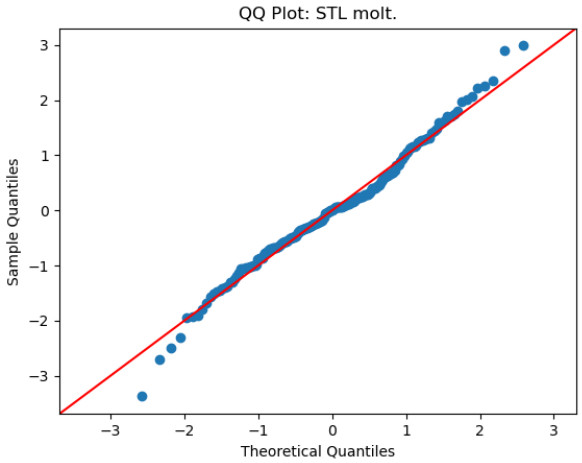


Figura 14

4 Predizione

In questa sezione mi occupo di fornire una previsione per il consumo di energia in Grecia per il periodo che va dal novembre 2024 all'ottobre 2025. In prima istanza verranno presi in considerazione quattro modelli: Holt-Winters additivo, moltiplicativo, metodo autoregressivo Yule-Walker e OLS. Per ognuno di questi modelli considero 15 training-set: l' $i-$esimo training-set è formato dai primi $n-16+i$ valori della serie, dove $n=202$ è il numero di unità temporali della serie. Su ciascun training-set viene addestrato il modello e successivamente calcolato l'errore sulla predizione del $n-15+i$-esimo valore, che non fa parte del training-set. La somma di questi errori su tutti e 15 i training set è l'errore di validazione. Alla fine ho scelto il modello con errore di validazione minore.

4.1 Holt-Winters

Per la previsione ho considerato il modello di Smorzamento esponenziale con trend e stagionalità, sia per il modello moltiplicativo che per quello additivo. Questi modelli sono parametrizzati da:

- α, β, γ, parametri che calibrano l'importanza delle informazioni "presenti" e "passate"
- condizioni iniziali, come intercetta, coefficiente angolare iniziale e previsioni per il primo periodo.

Per non appesantire eccessivamente il calcolo dei paremetri, ho cercato solo α, β, γ, le condizioni iniziali sono impostate di default dal programma. Per cercare i parametri ottimali vengono minimizzati gli scarti quadratici tramite la fuzione python `ExponentialSmoothing`, ottenendo i seguenti valori:

- **Additivo:** $\alpha = 0.5239$, $\beta = 0.0423$, $\gamma = 0.0453$.
- **Moltiplicativo:** $\alpha = 0.6061$, $\beta = 0.0001$, $\gamma = 0.0875$.

Poiché c'è il rischio che l'ottimizzazione fornisca dei minimi locali, effettuo, per entrambi i metodi, grid-research, cercando i parametri migliori, scelti tra l'insieme $\{n/10 : n \in [10]\}$ unito ai parametri trovati in precedenza. Ho ottenuto i seguenti risultati:

- **Additivo:** $\alpha = 0.1$, $\beta = 0.0423$, $\gamma = 0.0453$.
- **Moltiplicativo:** $\alpha = 0.1$, $\beta = 0.0001$, $\gamma = 0.6$.

Facendo validazione, spiegata come all'inizio della sezione, ottengo i seguenti errori:

- **Additivo:** 1180000.
- **Moltiplicativo:** 1450000.

Un altro confronto tra i due modelli è fornito dal seguente grafico, che mostra le predizioni dei due metodi paragonate con la serie originale. Per molte unità i due metodi si sovrappongono, mentre in alcune hanno una differenza più marcata.

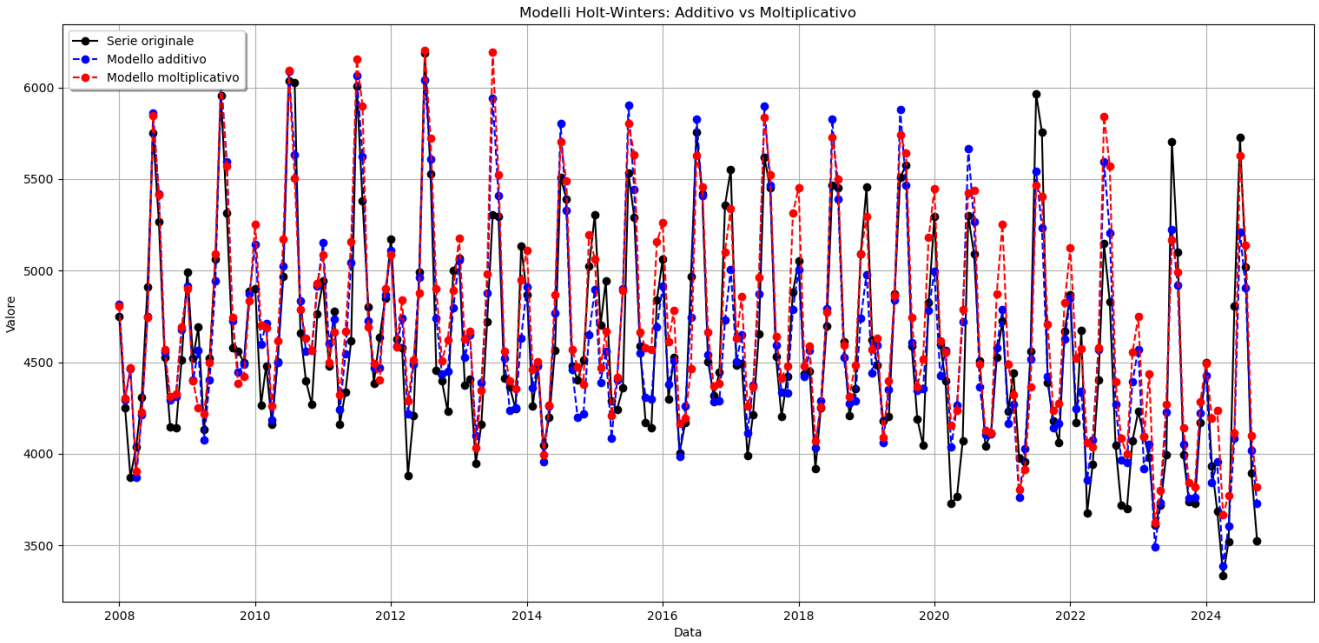


Figura 15

Prima di iniziare l'analisi dei metodi autoregressivi, è utile fornire un'analisi dei residui per capire se è possibile utilizzare intervalli parametrici per dare una stima nell'incertezza delle previsioni. Dai seguenti grafici non si nota una particolare struttura nei residui.

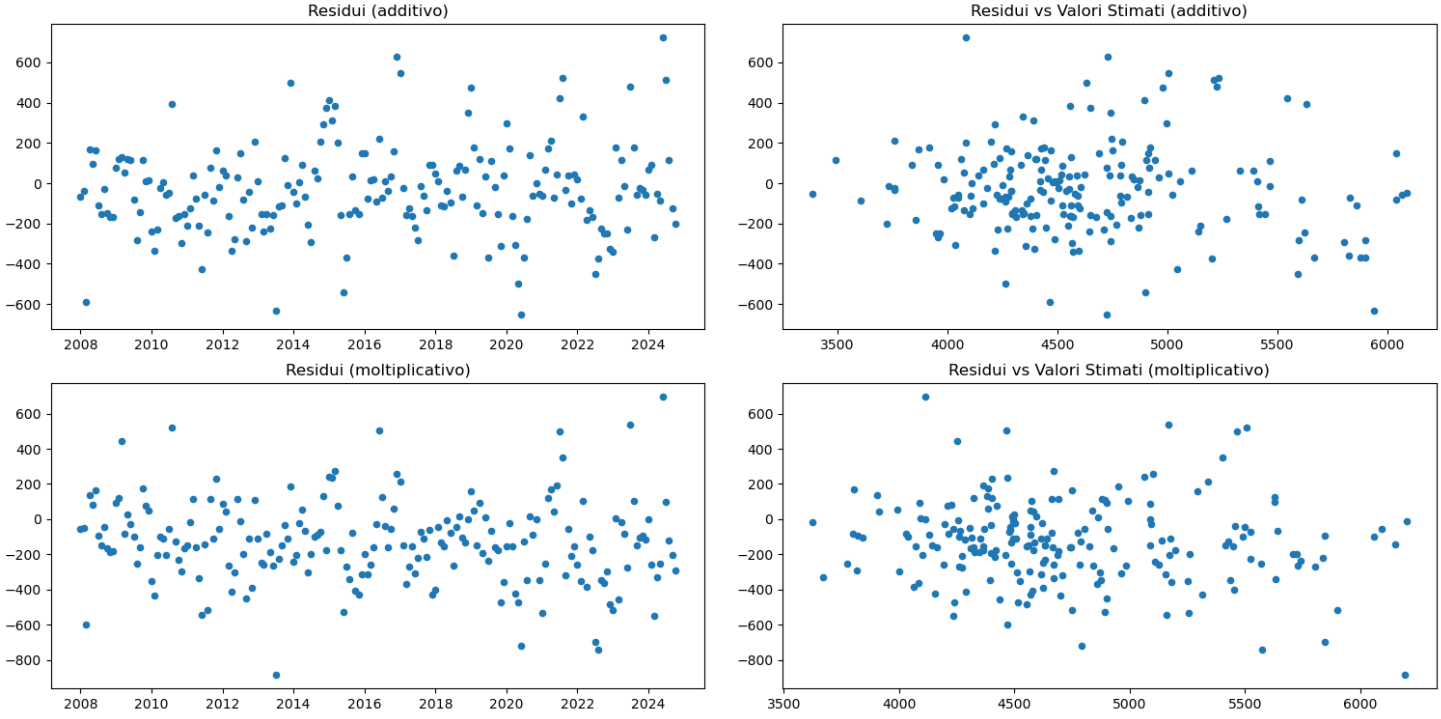


Figura 16

La mancanza di struttura è confermata per il modello moltiplicativo dal grafico seguente che mostra l' ACF dei residui, che diventa statisticamente nulla dopo pochi lag. Per il modello additivo i risultati non sono dissimili, ma è necessario osservare che l'ACF su alcuni lag è al limite della banda di incertezza.

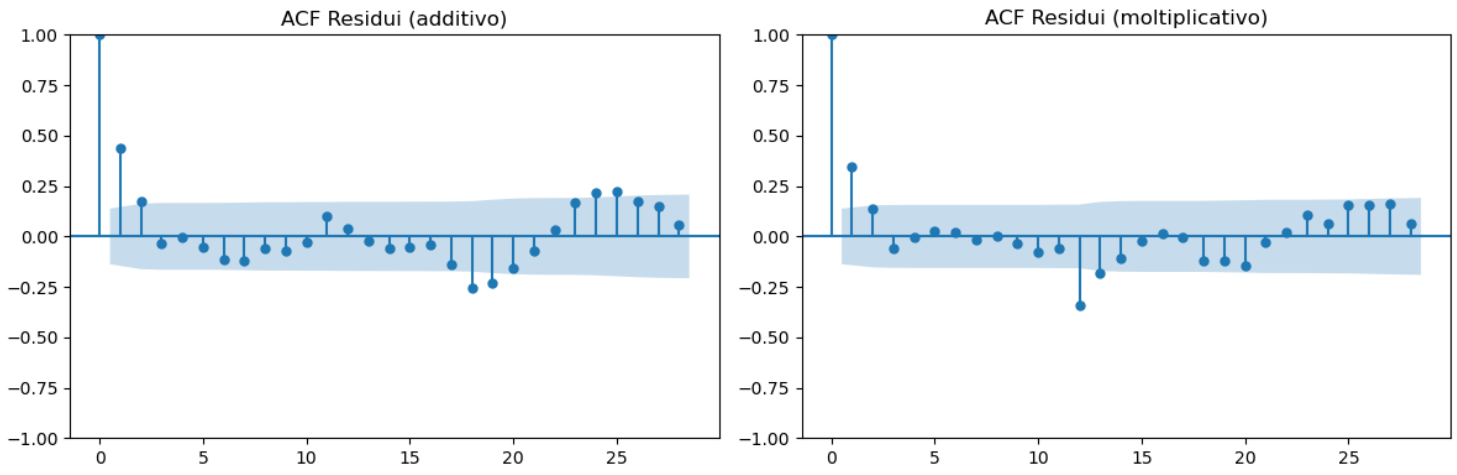


Figura 17

Valuto se i residui seguono una distribuzione gaussiana. I seguenti grafici mostrano le distribuzioni dei residui (in blu e con istogramma) confrontati con la distribuzione di una gaussiana con stessa media e varianza (in rosso) e il confronto tra i quantili empirici dei residui e quelli teorici di una guassiana con stessa media e varianza.

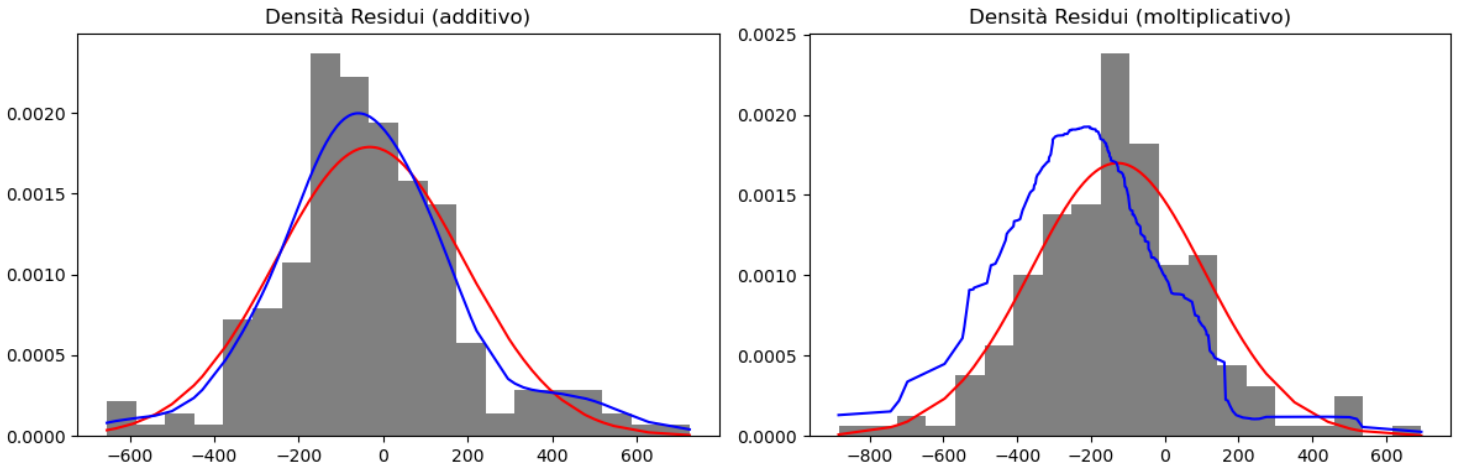


Figura 18

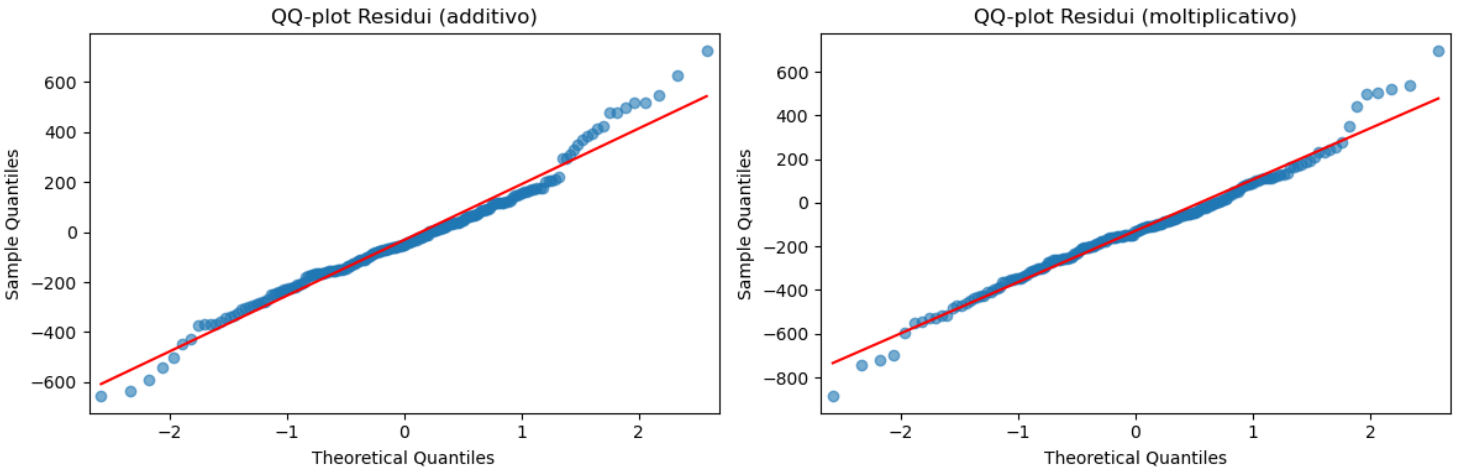


Figura 19

I grafici delle distribuzioni fanno credere che l'ipotesi di gaussianità sia poco probabile. Inoltre osservando i grafici dei quantili, notiamo che i quantili dei residui si discostano da quelli gaussiani, specialmente nel modello additivo. I sospetti sono confermati da dati quantitativi: il test di Shaprio-Wilk al livello 95% fa rifiutare l'ipotesi di gaussianità in entrambi i casi, e anche kurtosi e skewness non sono vicine a 0 (Tabella 1). In conclusione, nel caso dovessi usare uno di questi due metodi per la previsione, non verranno considerati intervalli di incertezza parametrici in quanto i residui non seguono una distribuzione gaussiana.

	skewness	kurtosis
additivo	0.390124	1.068847
moltiplicativo	0.207678	1.246838

Tabella 1

4.2 Metodi autoregressivi

In questa sezione verranno presi in esame i metodi autoregressivi ai minimi quadrati e Yule-Walker, cioè senza termine di intercetta. Per capire quanti termini temporali è necessario considerare nella regressione, osserviamo la funzione di autocorrelazione parziale:

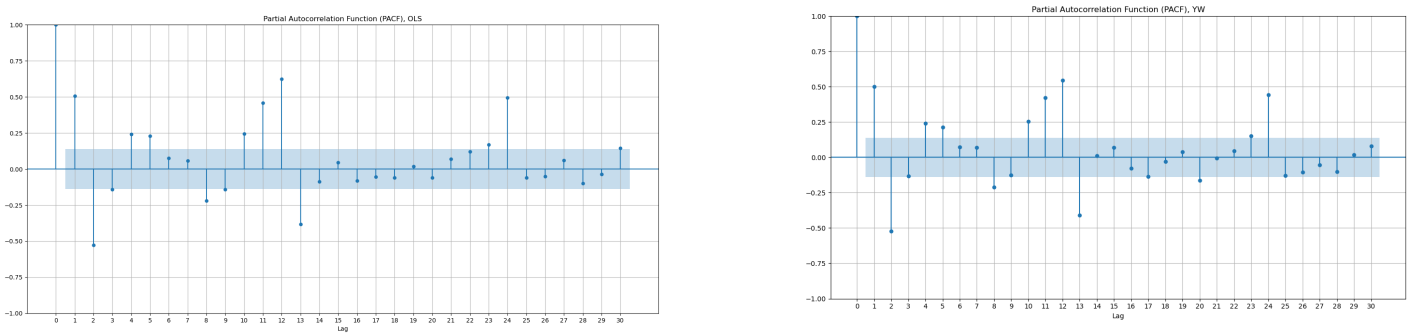


Figura 20

Dal grafico sembra che in entrambi casi il valore più estremo che consideriamo con parametro di input della regressione è la serie traslata di 24 lag, dato che 24 è l'ultimo valore di lag, almeno fino a 30 lag, con PACF statisticamente non nulla.
Prima di calcolare l'errore di validazione riduco il modello tramite il metodo del p-value: tolgo iterativamente dal modello di regressione lineare l'attributo con p-value maggiore fino a che tutte le features hanno p-value minore di 0.05. Da questa analisi emerge che i lag più rilevanti per il modello di Yule-Walker sono 1, 7, 12, 13, 19 e 24, mentre per OLS sono 1, 12, 13, 14, 24 lag. Dopo aver fatto ciò calcolo l'errore di validazione come spiegato all'inizio della sezione, ottendendo i seguenti risultati:

- **Modello YW:** errore: 1540000
- **Modello OLS:** errore: 1530000

In seguito vengono riportati in grafico i residui dei due modelli e il grafico dell'ACF per i residui dei due modelli. Dalla Figura 21, non si notato significative differenze rispetto ai residui di Holt-Winters. I risultati per i due modelli autoregressivi non sembrano differire molto, probabilmente perché la serie presenta una stagionalità stazionaria. Dal grafico dell'ACF (Figura 22) si osserva che i residui dei due modelli hanno perso totalmente struttura, un risultato leggermente migliore di smorzamento esponenziale. Tuttavia, osservando l'errore di validazione, decido di scegliere Holt-Winers additivo per la previsione.

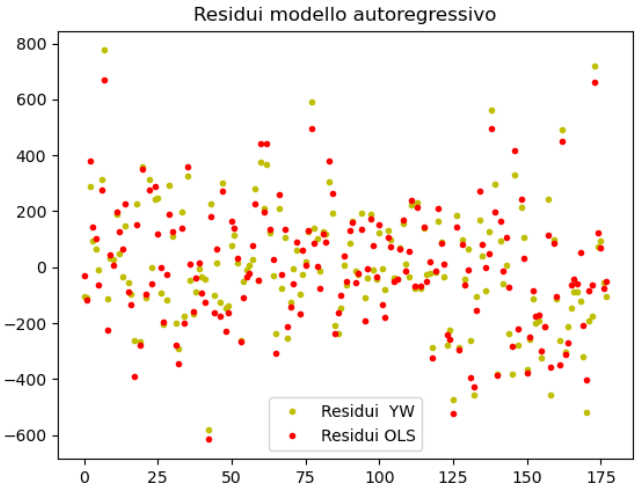


Figura 21

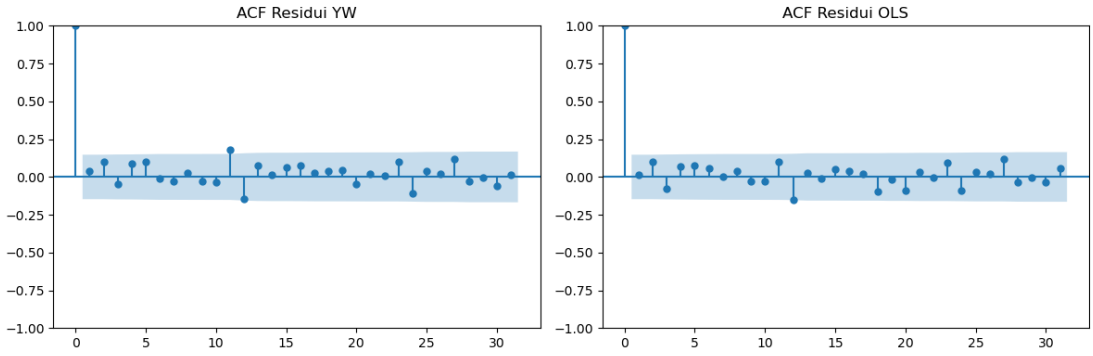


Figura 22

4.3 Previsione finale

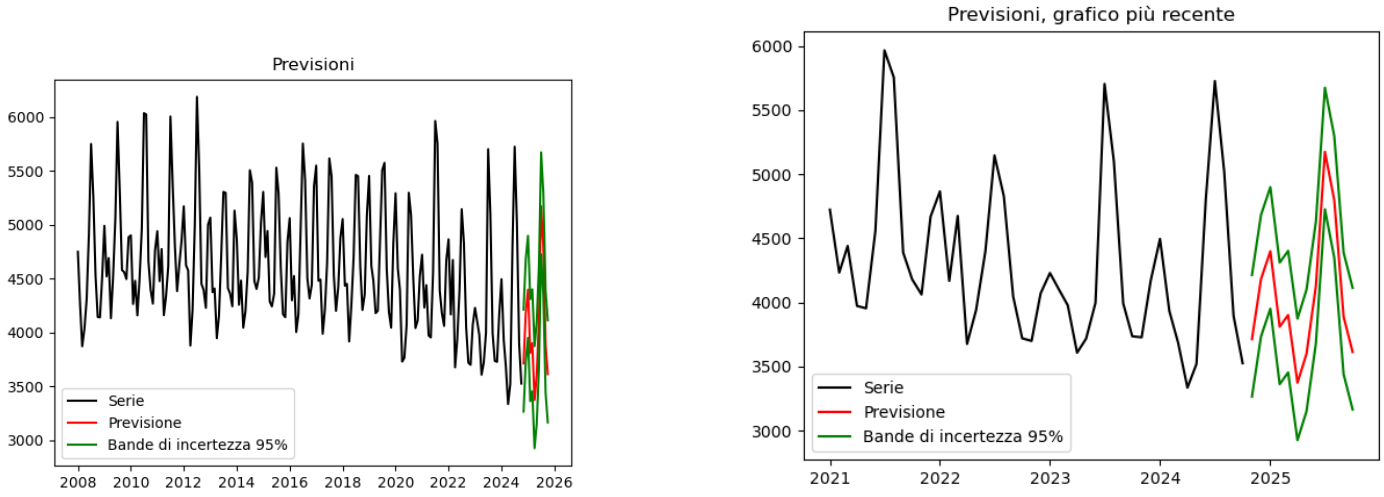


Figura 23

Uso Holt-Winters additivo per fare la previsione. L'algoritmo prevede iterativamente un mese alla volta: la prima previsione utilizza solo i dati della serie, mentre le altre utilizzano anche le previsioni precedenti.
Nei grafici viene riportata la serie in nero, in rosso la previsione per il periodo novembre 2024-ottobre 2025, in verde le bande di incertezza al 95%, calcolate in modo non parametrico osservando i quantili dei residui: le bande di incertezza vengono individuate traslando la previsione del 0.025-quantile e del 0.975-quantile dei residui. Infatti dall'analisi precedente è emerso che i residui non seguono una distribuzione gaussiana, nè che seguono una qualche distribuzione nota.

5 Conclusione

Alla fine la serie analizzata presenta una forte componente stagionale di 12 mesi, con stagionalità debolmente stazionaria. La componente di trend ha subito un mutamento intorno al 2020, non così sorprendente tenendo conto della pandemia e della crisi energetica che ha colpito l'Europa negli anni recenti.
Per quanto riguarda la previsione, smorzamento esponenziale con trend e stagionalità si è rivelato il modello con un errore di validazione minore, nonostante secondo l'ACF i residui hanno mantenuto una leggera struttura, non presente negli altri modelli. La banda di incertezza è ampia $948GWh$, e indica che nell'atteso picco estivo del 2025 il consumo energetico dovrebbe essere minore o uguale a quello dello scorso anno, dunque per l'azienda energetica è atteso un consumo di risorse minore, quindi potrebbe valutare di fare dei tagli sulle importazioni per il prossimo anno o almeno di evitare di fare investimenti in senso espansivo.

A Codice

Listing 1: Analisi della serie

```
 import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
from statsmodels.tsa.stattools import acf
from scipy.stats import norm
import statsmodels.api as sm
from scipy import stats as st
from statsmodels.tsa.seasonal import STL

tabella = pd.read_csv("tabella.csv")
tabella['TIME_PERIOD'] = pd.to_datetime(tabella['TIME_PERIOD'])
tabella.set_index('TIME_PERIOD', inplace=True)
tabella.sort_index(inplace=True)
tabella = tabella["OBS_VALUE"]
print(tabella.tail())

plt.plot(tabella,c='b')
plt.title("Grafico della serie")
plt.xlabel("Tempo")
plt.ylabel("Consumo (GWh)")
plt.grid(True)
plt.show()


stl = STL(tabella, seasonal=7) # Crea l'oggetto STL
result = stl.fit()
seasonal_component = result.seasonal

#cogliamo la stagionalit

diff_tp = tabella.diff().dropna()

plt.figure(figsize=(10, 5))
plt.plot(diff_tp, c='b')
plt.title('Serie delle differenze')
plt.xlabel('Tempo')
plt.ylabel('Differenze')
plt.grid(True)
plt.show()


fig, ax = plt.subplots()
plot_acf(tabella, lags=30, ax=ax)
ax.set_xticks(range(0, 31, 1))
ax.set_xlabel('Lag')
ax.grid(True)
plt.title('Autocorrelation Function (ACF)')
plt.show()

fig, ax = plt.subplots()
plot_acf(diff_tp, lags=30, ax=ax)
ax.set_xticks(range(0, 31, 1))
ax.set_xlabel('Lag')
ax.grid(True)
plt.title('ACF delle differenze')
plt.show()

tabella_sim = pd.concat([tabella, pd.Series([np.nan, np.nan])], ignore_index=True)
tabella_sim = tabella_sim.values.reshape(-1,12)
tabella_sim = tabella_sim.transpose()
plt.style.use('dark_background')
colors = plt.cm.hot(np.linspace(0, 1, 17))
for i in range(tabella_sim.shape[1]):
    plt.plot(range(1, 13), tabella_sim[:, i]-(tabella_sim[:,i]).mean(), color=colors[i])
plt.title("Andamento mensile centrato")
plt.show()
for i in range(tabella_sim.shape[1]):
    plt.plot(range(1, 13), tabella_sim[:, i], color=colors[i])
medie = np.nanmean(tabella_sim, axis=1)
print(medie)
plt.plot(range(1, 13), medie, marker='o', linestyle='-', linewidth=5, color='white', label='Media annuale')
sd_tab = np.nanstd(tabella_sim, axis=1)
plt.plot(range(1, 13), medie + sd_tab, marker='o', linestyle='-', linewidth=5, color='gray', label='Media + Dev.
    St.')
plt.plot(range(1, 13), medie - sd_tab, marker='o', linestyle='-', linewidth=5, color='gray', label='Media - Dev.
    St.')
```

```
plt.xlabel("Mesi")
plt.ylabel("Valori")
plt.title("Andamenti mensili e media annuale")
plt.legend()
plt.grid(color='gray', linestyle='--', linewidth=0.5)
plt.show()
plt.style.use('default')


#decidiamo se modello moltiplicativo o additivo

result_a = seasonal_decompose(tabella, model='additive', period=12)
fig = result_a.plot()
for ax in fig.axes:
    ax.set_title("")
fig.suptitle("Decomposizione additiva")
plt.show()

result_m = seasonal_decompose(tabella, model='multiplicative', period=12)
fig = result_m.plot()
for ax in fig.axes:
    ax.set_title("")
fig.suptitle("Decompodizione moltiplicativa")
plt.show()

stl = STL(tabella, seasonal=7)
result = stl.fit()
fig = result.plot()
for ax in fig.axes:
    ax.set_title("")
fig.suptitle("Decomposizione STL")
plt.show()

stl = STL(np.log(tabella), seasonal=7)
result = stl.fit()
fig, axes = plt.subplots(4, 1, figsize=(10, 8), sharex=True)

# Componente osservata
axes[0].plot(np.exp(result.observed), label="Observed")
axes[0].set_ylabel("Observed")
axes[0].ticklabel_format(axis="y", style="sci", scilimits=(0, 0)) # Attiva notazione scientifica
axes[0].legend()

# Componente trend
axes[1].plot(np.exp(result.trend), label="Trend", color="orange")
axes[1].set_ylabel("Trend")
axes[1].ticklabel_format(axis="y", style="sci", scilimits=(0, 0))
axes[1].legend()

# Componente stagionale
axes[2].plot(np.exp(result.seasonal), label="Seasonal", color="green")
axes[2].set_ylabel("Seasonal")
axes[2].ticklabel_format(axis="y", style="sci", scilimits=(0, 0))
axes[2].legend()

# Componente residuo
axes[3].plot(np.exp(result.resid), label="Residuals", color="red")
axes[3].set_ylabel("Residuals")
axes[3].ticklabel_format(axis="y", style="sci", scilimits=(0, 0))
axes[3].legend()

# Label asse x
axes[3].set_xlabel("Time")

# Mostra il grafico
plt.tight_layout()
fig.suptitle("Decomposizione STL moltiplicativa")
plt.show()

plt.plot(result_a.seasonal, c="k", label="Additivo")
plt.plot(np.nanmean(result_m.trend)*(result_m.seasonal-1), c='r',label="Moltiplicativo")
plt.title("Confronto stagionalit")
plt.legend()
```

```
plt.show()

plt.plot(result_a.resid, c="k", label="Additivo")
plt.plot(np.nanmean(result_m.trend)*(result_m.resid-1), c='r', label="Moltiplicativo")
plt.title("Confronto residui")
plt.legend()
plt.show()

#analisi dei residui
valori = tabella.values[6:-6]
aresid = result_a.resid.values
aresid = aresid[~np.isnan(aresid)]
mresid = result_m.resid.values
mresid = mresid[~np.isnan(mresid)]
plt.xlabel("Unit temporali")
plt.plot(aresid,'.')
plt.title("Residui additivi")
plt.show()
fig, ax = plt.subplots()
plot_acf(aresid, lags=30, ax=ax)
ax.set_xticks(range(0, 31, 1))
ax.set_xlabel('Lag')
ax.grid(True)
plt.title('ACF residui additivi')
plt.show()

fig, ax = plt.subplots()
plot_acf(mresid, lags=30, ax=ax)
ax.set_xticks(range(0, 31, 1))
ax.set_xlabel('Lag')
ax.grid(True)
plt.title('ACF residui moltiplicativi')
plt.show()


plt.plot(mresid,'.')
plt.title("Residui moltiplicativi")
plt.xlabel("Unit temporali")
plt.show()

acf_values = acf(aresid, fft=True)
acf_std = np.std(acf_values)
print("Deviazione standard dell'ACF:", acf_std)
acf_values = acf(np.log(mresid), fft=True)
acf_std = np.std(acf_values)
print("Deviazione standard dell'ACF:", acf_std)
print("Var additiva: ", np.var(aresid)/np.var(valori))
print("Var molt: ", np.var(np.log(mresid))/np.var(np.log(valori)))

#valori simili: probabilmente c' l'effetto che sembra additivo perch approssimiamo al primo ordine

#vediamo se i residui seguono una distribuzione gaussiana
mresidl = mresid
x=np.linspace(min(mresidl), max(mresidl),1000)
plt.hist(mresidl,density=True, bins=40, label="Residui moltiplicativi")
plt.plot(x, norm.pdf(x,loc=np.mean(mresidl),scale=np.std(mresidl)), label="Gaussiana")
plt.title("Confronto distribuzione moltiplicativa e gaussiana")
plt.legend()
plt.show()
sm.qqplot((mresidl-np.mean(mresidl))/(np.std(mresidl)), line='45')
plt.title("QQ Plot: moltiplicativo")
plt.show()
ska = st.skew((aresid-np.mean(aresid))/np.std(aresid))
skm = st.skew((mresid-np.mean(mresid))/np.std(mresid))
ka = st.kurtosis((aresid-np.mean(aresid))/np.std(aresid))
km = st.kurtosis((aresid-np.mean(aresid))/np.std(aresid))
print(st.shapiro(mresidl))
print("Skewness_add: ", ska)
print("Skewness_molt: ", skm)
print("Kurtosi_add: ", ka)
print("Kurtosi_molt: ", km)

x=np.linspace(min(aresid), max(aresid),1000)
```

```
plt.hist(aresid,density=True, bins=40, label="Residui additivi")
plt.plot(x, norm.pdf(x,loc=np.mean(aresid),scale=np.std(aresid)), label="Gaussiana")
plt.title("Confronto distribuzione additiva e gaussiana")
plt.legend()
plt.show()
sm.qqplot((aresid-np.mean(aresid))/(np.std(aresid)), line='45')
plt.title("QQ Plot: additivo")
plt.show()

print(st.shapiro(aresid))


stl = STL(tabella, seasonal=7)
result = stl.fit()
resid =result.resid.values

plt.xlabel("Unit temporali")
plt.plot(resid,'.')
plt.title("Residui STL")
plt.show()

fig, ax = plt.subplots()
plot_acf(resid, lags=30, ax=ax)
ax.set_xticks(range(0, 31, 1))
ax.set_xlabel('Lag')
ax.grid(True)
plt.title('ACF residui STL')
plt.show()

acf_values = acf(resid, fft=True)
acf_std = np.std(acf_values)
print("Deviazione standard dell'ACF:", acf_std)
print("Var STL: ", np.var(resid)/np.var(valori))

x=np.linspace(min(resid), max(resid),1000)
plt.hist(resid,density=True, bins=40, label="Residui STL")
plt.plot(x, norm.pdf(x,loc=np.mean(resid),scale=np.std(resid)), label="Gaussiana")
plt.title("Confronto distribuzione STL e gaussiana")
plt.legend()
plt.show()
sm.qqplot((resid-np.mean(resid))/(np.std(resid)), line='45')
plt.title("QQ Plot: STL")
plt.show()

sk = st.skew((resid-np.mean(resid))/np.std(resid))
k = st.kurtosis((resid-np.mean(resid))/np.std(resid))
print("Skewness_stl: ", sk)
print("Kurtosi_stl: ", k)
print("Shaprio stl: ", st.shapiro(resid))

stl = STL(np.log(tabella), seasonal=7)
result = stl.fit()
resid =result.resid.values

plt.xlabel("Unit temporali")
plt.plot(resid,'.')
plt.title("Residui STL moltiplicaivo")
plt.show()

fig, ax = plt.subplots()
plot_acf(resid, lags=30, ax=ax)
ax.set_xticks(range(0, 31, 1))
ax.set_xlabel('Lag')
ax.grid(True)
plt.title('ACF residui STL moltiplicativi')
plt.show()

acf_values = acf(resid, fft=True)
acf_std = np.std(acf_values)
print("Deviazione standard dell'ACF:", acf_std)
print("Var STL molt.: ", np.var(resid)/np.var(np.log(valori)))

x=np.linspace(min(resid), max(resid),1000)
```

```
plt.hist(resid,density=True, bins=40, label="Residui STL molt.")
plt.plot(x, norm.pdf(x,loc=np.mean(resid),scale=np.std(resid)), label="Gaussiana")
plt.title("Confronto distribuzione STL molt. e gaussiana")
plt.legend()
plt.show()
sm.qqplot((resid-np.mean(resid))/(np.std(resid)), line='45')
plt.title("QQ Plot: STL molt.")
plt.show()
sk = st.skew((resid-np.mean(resid))/np.std(resid))
k = st.kurtosis((resid-np.mean(resid))/np.std(resid))
print("Skewness_stl_molt: ", sk)
print("Kurtosi_stl_molt: ", k)
print("Shaprio stl molt: ", st.shapiro(resid))
```

Listing 2: Previsione

```
  import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.regression.linear_model import yule_walker

from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import acf
from scipy.stats import norm
import statsmodels.api as sm
from scipy import stats as st
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from scipy.stats import shapiro, skew, kurtosis, norm
from scipy.stats import gaussian_kde




tabella = pd.read_csv("tabella.csv")
tabella['TIME_PERIOD'] = pd.to_datetime(tabella['TIME_PERIOD'])
tabella.set_index('TIME_PERIOD', inplace=True)
tabella.sort_index(inplace=True)
tabella.index = tabella.index.to_period('M').to_timestamp()
tabella = tabella["OBS_VALUE"]

#holt-winters

amodel = ExponentialSmoothing(tabella, trend='add', seasonal='add', seasonal_periods=12)
hwa = amodel.fit()
alpha = hwa.model.params["smoothing_level"]
beta = hwa.model.params["smoothing_trend"]
gamma = hwa.model.params["smoothing_seasonal"]
print(f"additivo - alpha: {alpha:.4f}, beta: {beta:.4f}, gamma: {gamma:.4f}")

mmodel = ExponentialSmoothing(tabella, trend='add', seasonal='mul', seasonal_periods=12)
hwm = mmodel.fit()

alpha = hwm.model.params["smoothing_level"]
beta = hwm.model.params["smoothing_trend"]
gamma = hwm.model.params["smoothing_seasonal"]
print(f"molti - alpha: {alpha:.4f}, beta: {beta:.4f}, gamma: {gamma:.4f}")

nt = 15 #numero di test set
ft = 1 #unit di tempo a cui estendnere la previsione
n = tabella.shape[0]
idt = tabella.index[0]
fdt = tabella.index[-1]
pdt = 12

max = np.array([1e+10,0,0,0])
```

```
#cerco i parametri tramite grid-search
a =np.arange(1,10)/10
a = np.concatenate(([0.5239],a))
b =np.arange(1,10)/10
b = np.concatenate(([0.0423],b))
c =np.arange(1,10)/10
c = np.concatenate(([0.0453],c))

for i in a:
    print(i)
    for j in b:
        for k in c:
            err =0

            for l in np.arange(n-nt-ft,n-ft):
                index_1 = tabella.index[l-1]
                index_2= tabella.index[l]
                train = tabella[idt:index_1]
                index_3 = tabella.index[l+ft-1]
                test = tabella[index_2:index_3]
                model_hw = ExponentialSmoothing(train, trend='add', seasonal='add', seasonal_periods=12)
                train.hw = model_hw.fit(smoothing_level=i, smoothing_trend=j, smoothing_seasonal=k, optimized= False)
                predictions = train.hw.forecast(steps=ft)
                err += np.sum((test.values - predictions.values) ** 2)
            if (max[0]>err):
                    max[0]=err
                    max[1]=i
                    max[2]=j
                    max[3]=k

print ("errore add: ",max[0], "alpha: ", max[1], "beta", max[2], "gamma", max[3] )

#cerco i parametri tramite grid-search
a =np.arange(1,10)/10
a = np.concatenate(([0.6061],a))
b =np.arange(1,10)/10
b = np.concatenate(([0.0001],b))
c =np.arange(1,10)/10
c = np.concatenate(([0.0875],c))

for i in a:
    print(i)
    for j in b:
        for k in c:
            err =0

            for l in np.arange(n-nt-ft,n-ft):
                index_1 = tabella.index[l-1]
                index_2= tabella.index[l]
                train = tabella[idt:index_1]
                index_3 = tabella.index[l+ft-1]
                test = tabella[index_2:index_3]
                model_hw = ExponentialSmoothing(train, trend='add', seasonal='mul', seasonal_periods=12)
                train.hw = model_hw.fit(smoothing_level=i, smoothing_trend=j, smoothing_seasonal=k, optimized= False)
                predictions = train.hw.forecast(steps=ft)
                err += np.sum((test.values - predictions.values) ** 2)
            if (max[0]>err):
                    max[0]=err
                    max[1]=i
                    max[2]=j
                    max[3]=k

print ("errore add: ",max[0], "alpha: ", max[1], "beta", max[2], "gamma", max[3] )

model_hwa = ExponentialSmoothing(tabella, trend='add', seasonal='add', seasonal_periods=12)
hwa = model_hwa.fit(smoothing_level=0.1, smoothing_trend=0.0423, smoothing_seasonal=0.0453, optimized=False)
model_hwm = ExponentialSmoothing(tabella, trend='add', seasonal='mul', seasonal_periods=12)
hwm = model_hwm.fit(smoothing_level=0.1, smoothing_trend=0.0001, smoothing_seasonal=0.6, optimized=False)

fitted_hwa = hwa.fittedvalues
fitted_hwm = hwm.fittedvalues

plt.figure(figsize=(10, 6))
```

```
plt.plot(tabella, label="Serie originale", color="black", marker="o")
plt.plot(fitted_hwa, label="Modello additivo", color="blue", linestyle="--", marker="o")
plt.plot(fitted_hwm, label="Modello moltiplicativo", color="red", linestyle="--", marker="o")

# Legenda
plt.legend(loc="upper left", fontsize=10, frameon=True, fancybox=True, shadow=True)
plt.title("Modelli Holt-Winters: Additivo vs Moltiplicativo")
plt.xlabel("Data")
plt.ylabel("Valore")
plt.grid()
plt.show()

# Estrazione dei residui
hwa_r = tabella - hwa.fittedvalues
hwm_r = tabella - hwm.fittedvalues

# Proporzione di varianza non spiegata
var_ratio_hwa = hwa_r.var() / tabella.var()
var_ratio_hwm = hwm_r.var() / tabella.var()

print("Proporzione di varianza non spiegata (additivo):", var_ratio_hwa)
print("Proporzione di varianza non spiegata (moltiplicativo):", var_ratio_hwm)

# Indicatori di forma (skewness e kurtosis)
fm = pd.DataFrame({
    "skewness": [skew(hwa_r), skew(hwm_r)],
    "kurtosis": [kurtosis(hwa_r), kurtosis(hwm_r)]
}, index=["additivo", "moltiplicativo"])

print("\nIndicatori di forma:")
print(fm)

# Scatterplot
fig, axes = plt.subplots(2, 2, figsize=(12, 8))

axes[0, 0].scatter(hwa_r.index, hwa_r, s=20)
axes[0, 0].set_title("Residui (additivo)")
axes[0, 1].scatter(hwa.fittedvalues, hwa_r, s=20)
axes[0, 1].set_title("Residui vs Valori Stimati (additivo)")

axes[1, 0].scatter(hwm_r.index, hwm_r, s=20)
axes[1, 0].set_title("Residui (moltiplicativo)")
axes[1, 1].scatter(hwm.fittedvalues, hwm_r, s=20)
axes[1, 1].set_title("Residui vs Valori Stimati (moltiplicativo)")

plt.tight_layout()
plt.show()

# Autocorrelazione
fig, axes = plt.subplots(1, 2, figsize=(12, 4))

plot_acf(hwa_r, lags=28, ax=axes[0])
axes[0].set_title("ACF Residui (additivo)")

plot_acf(hwm_r, lags=28, ax=axes[1])
axes[1].set_title("ACF Residui (moltiplicativo)")

plt.tight_layout()
plt.show()

#  Densit  empirica
fig, axes = plt.subplots(1, 2, figsize=(12, 4))

# Additivo
kde = gaussian_kde(hwa_r)

axes[0].hist(hwa_r, bins=20, density=True, color="gray")
axes[0].plot(np.sort(hwa_r), norm.pdf(np.sort(hwa_r), np.mean(hwa_r), np.std(hwa_r)), color="red")
axes[0].plot(np.sort(hwa_r), kde(np.sort(hwa_r)), color="blue")
axes[0].set_title("Densit Residui (additivo)")

# Moltiplicativo
kde = gaussian_kde(hwm_r)
```

```
axes[1].hist(hwm_r, bins=20, density=True, color="gray")
axes[1].plot(np.sort(hwm_r), norm.pdf(np.sort(hwm_r), np.mean(hwm_r), np.std(hwm_r)), color="red")
axes[1].plot(np.sort(hwm_r), kde(np.sort(hwa_r)), color="blue")
axes[1].set_title("Densit Residui (moltiplicativo)")

plt.tight_layout()
plt.show()

# Grafico quantile-quantile
fig, axes = plt.subplots(1, 2, figsize=(12, 4))

sm.qqplot(hwa_r, line="s", ax=axes[0], marker="o", alpha=0.6)
axes[0].set_title("QQ-plot Residui (additivo)")

sm.qqplot(hwm_r, line="s", ax=axes[1], marker="o", alpha=0.6)
axes[1].set_title("QQ-plot Residui (moltiplicativo)")

plt.tight_layout()
plt.show()

# Test di Gaussianit
shapiro_hwa = shapiro(hwa_r)
shapiro_hwm = shapiro(hwm_r)

print("\nTest di Gaussianit (Shapiro-Wilk):")
print(f"Residui Additivo: p-value = {shapiro_hwa.pvalue:.4f}")
print(f"Residui Moltiplicativo: p-value = {shapiro_hwm.pvalue:.4f}")


#autoregressivo diretto yw
fig, ax = plt.subplots()
plot_acf(tabella, lags=30, ax=ax)
ax.set_xticks(range(0, 31, 1))
ax.set_xlabel('Lag')
ax.grid(True)
plt.title('Autocorrelation Function (ACF)')
plt.show()

fig, ax = plt.subplots()
plot_pacf(tabella, lags=30, ax=ax, method="ols")
ax.set_xticks(range(0, 31, 1))
ax.set_xlabel('Lag')
ax.grid(True)
plt.title('Partial Autocorrelation Function (PACF), OLS')
plt.show()

fig, ax = plt.subplots()
plot_pacf(tabella, lags=30, ax=ax, method="yw")
ax.set_xticks(range(0, 31, 1))
ax.set_xlabel('Lag')
ax.grid(True)
plt.title('Partial Autocorrelation Function (PACF), YW')
plt.show()

L = len(tabella)
l = 24

lm =LinearRegression()

mnt = np.zeros((L - l, l + 1))
for i in range(1, l + 2):
    mnt[:, i - 1] = tabella[i - 1:L - l - 1 + i]
mnt_df = pd.DataFrame(mnt, columns=[f"X{i}" for i in range(1, l + 2)])
y = mnt_df["X25"]
features= mnt_df[mnt_df.columns.difference(["X25"])]
#features = sm.add_constant(features), togliere commento se si vuole ridurre OLS
modello = sm.OLS(y, features).fit()

while ((modello.pvalues>0.05).any()):
   colonna_drop = modello.pvalues.idxmax()
   features = features.drop(columns=colonna_drop)
   modello = sm.OLS(y, features).fit()
```

```
print(modello.summary())



# Parametri
nt = 15 # Numero di test set
ft = 1  # Passi futuri da prevedere
n = len(tabella)
pdt = 12 # Periodo della serie (stagionalit)
li = 24 # Numero massimo di lag per il modello autoregressivo

err_lmr = np.zeros(nt)
err_ols = np.zeros(nt)
err_hwa = np.zeros(nt)
err_hwm = np.zeros(nt)

# Ciclo di validazione
for l in range(n - nt - ft, n - ft):
    # Costruzione di train e test
    train = tabella.iloc[:l]
    test = tabella.iloc[l:l + ft]

    # Verifica della lunghezza minima del train set
    if len(train) <= li:
        continue

    # Modello autoregressivo ridotto
    L = len(train)
    mtrain = np.zeros((L - li, li + 1))

    for i in range(0, li + 1):
        start_idx = i
        end_idx = L - li + i
        mtrain[:, i ] = train.iloc[start_idx:end_idx].values

    # Creazione del DataFrame
    mtrain_df = pd.DataFrame(mtrain, columns=[f"X{i}" for i in range(1, li + 2)])
    y_train = mtrain_df["X25"]
    X_train = mtrain_df[["X1", "X7", "X12", "X13", "X19", "X24"]]

    # Regressione lineare senza costanti
    modello = sm.OLS(y_train, X_train).fit()

    # Previsioni con il modello lineare
    train_lmr_p = np.zeros(L + ft)
    train_lmr_p[:L] = train.values

    for i in range(0, ft):
        x_lag = np.array([train_lmr_p[L + i - 24], train_lmr_p[L + i - 18],
                          train_lmr_p[L + i - 13], train_lmr_p[L + i - 12],
                          train_lmr_p[L + i - 6], train_lmr_p[L + i - 1]])
        # Previsione
        train_lmr_p[L + i] = modello.predict(x_lag.reshape(1,-1))[0]

    # Calcolo dell'errore

    err_lmr[l - (n - nt - ft)] = test.values[-1] - train_lmr_p[L + ft-1]

    # Modello autoregressivo OLS
    if len(train) > 25: # Verifica che il training set sia sufficiente
        model_hwa = ExponentialSmoothing(train, trend='add', seasonal='add', seasonal_periods=12)
        hwa = model_hwa.fit(smoothing_level=0.1, smoothing_trend=0.0423, smoothing_seasonal=0.0453, optimized=False)
        model_hwm = ExponentialSmoothing(train, trend='add', seasonal='mul', seasonal_periods=12)
        hwm = model_hwm.fit(smoothing_level=0.1, smoothing_trend=0.0001, smoothing_seasonal=0.6, optimized=False)
        model_ols = AutoReg(train, lags=[1,12,13,14,24], old_names=False).fit()

        pred_hwa = hwa.predict(start=len(train), end=len(train) + ft - 1)
        pred_hwm = hwm.predict(start=len(train), end=len(train) + ft - 1)

        pred_ols = model_ols.predict(start=len(train), end=len(train) + ft - 1)
        err_ols[l - (n - nt - ft)] = test.values[-1] - pred_ols.values[-1]
```

```
        err_hwa[l - (n - nt - ft)] = test.values[-1] - pred_hwa.values[-1]
        err_hwm[l - (n - nt - ft)] = test.values[-1] - pred_hwm.values[-1]

# Errori totali
print(f"Modello autoregressivo ridotto - errore: {np.sum(err_lmr ** 2):.4f}")
print(f"Modello autoregressivo OLS - errore: {np.sum(err_ols ** 2):.4f}")
print(f"HW additivo - errore: {np.sum(err_hwa ** 2):.4f}")
print(f"HW moltiplicativo- errore: {np.sum(err_hwm ** 2):.4f}")

y = mnt_df["X25"]

features= mnt_df[mnt_df.columns.difference(["X25"])]
features = features[["X1", "X7", "X12", "X13", "X19", "X24"]]
modello = sm.OLS(y, features).fit()
RES = modello.resid

features= mnt_df[mnt_df.columns.difference(["X25"])]
features = features[["X1", "X12", "X13", "X14", "X24"]]
features = sm.add_constant(features)
modello = sm.OLS(y, features).fit()
RES2 = modello.resid
plt.plot(RES,'.',c='y', label="Residui YW")
plt.plot(RES2,'.',c='r', label="Residui OLS")
plt.title("Residui modello autoregressivo")
plt.legend()
plt.show()

fig, axes = plt.subplots(1, 2, figsize=(12, 4))

plot_acf(RES, lags=31, ax=axes[0])
axes[0].set_title("ACF Residui YW")

plot_acf(RES2, lags=31, ax=axes[1])
axes[1].set_title("ACF Residui OLS")

plt.tight_layout()
plt.show()

#previsioni
model_hwa = ExponentialSmoothing(tabella, trend='add', seasonal='add', seasonal_periods=12)
hwa = model_hwa.fit(smoothing_level=0.1, smoothing_trend=0.0423, smoothing_seasonal=0.0453, optimized=False)
predictions = hwa.forecast(steps=12)
err = hwa.resid

plt.figure()
plt.plot(tabella, c='k', label="Serie")
plt.plot(predictions,c='r',label="Previsione")
plt.plot(predictions+np.quantile(err,0.975),c='g',label="Bande di incertezza 95%")
plt.plot(predictions+np.quantile(err,0.025),c='g')
plt.title("Previsioni")
plt.legend()
plt.show()

plt.figure()
plt.plot(tabella["2021":], c='k', label="Serie")
plt.plot(predictions,c='r', label="Previsione")
plt.plot(predictions+np.quantile(err,0.975),c='g', label="Bande di incertezza 95%")
plt.plot(predictions+np.quantile(err,0.025),c='g')
plt.title("Previsioni, grafico pi recente")
plt.legend()
plt.show()

print(np.quantile(err,0.975)-np.quantile(err,0.025))
```