Cos'è il modello SARIMA?

Il modello **SARIMA (Seasonal AutoRegressive Integrated Moving Average)** è una versione estesa del modello ARIMA che tiene conto della stagionalità nei dati. È utile quando i dati presentano una periodicità definita (ad esempio, mensile o annuale) e permette di cogliere sia il comportamento generale della serie temporale che le fluttuazioni stagionali.

La notazione generale è:

$$SARIMA(p,d,q)(P,D,Q,m)$$

- **(p, d, q)**: Parametri non stagionali.
    - **p**: Numero di termini autoregressivi (dipendenza dai valori precedenti).
    - **d**: Ordine della differenziazione (rende la serie stazionaria).
    - **q**: Numero di termini delle medie mobili (dipendenza dagli errori passati).
- **(P, D, Q, m)**: Parametri stagionali.
    - **P**: Numero di termini autoregressivi stagionali.
    - **D**: Ordine della differenziazione stagionale.
    - **Q**: Numero di termini delle medie mobili stagionali.
    - **m**: Lunghezza del ciclo stagionale (ad esempio 12 per i dati mensili).

Perché SARIMA per il budget dell'azienda?

Applicare SARIMA ai dati di vendita mensile della tua azienda ha diversi vantaggi:

- **Cattura la stagionalità**: L'abbigliamento è fortemente stagionale, con cicli ben definiti (Primavera/Estate e Autunno/Inverno).
- **Previsioni affidabili**: Il modello permette di generare previsioni accurate per futuri periodi, utili per pianificare il budget.
- **Gestione delle decisioni strategiche**: Aiuta la buyer a definire quantità e valore degli acquisti da fare circa un anno in anticipo, basandosi su una previsione solida.

Spiegazione passo-passo del programma

1. Caricamento e preparazione dei dati (`load_data()`)

Questa funzione:

- Legge i dati mensili di vendita da un file Excel.
- Estrae e converte il campo `"Mese Anno"` in formato datetime.
- Ordina e imposta la frequenza mensile (`MS` sta per "Month Start").

```
def load_data(file_path):
    df = pd.read_excel(file_path)
    # Conversione della colonna "Mese Anno" in data
    df[['Mese', 'Anno']] = df['Mese Anno'].str.split(' ', expand=True)
    df['Anno'] = pd.to_numeric(df['Anno'])

    mesi_mapping = {"Gennaio": 1, "Febbraio": 2, "Marzo": 3, "Aprile": 4, "Maggio": 5,
"Giugno": 6,
                    "Luglio": 7, "Agosto": 8, "Settembre": 9, "Ottobre": 10, "Novembre": 11,
"Dicembre": 12}
    df['Mese'] = df['Mese'].map(mesi_mapping)

    df['Data'] = pd.to_datetime({'year': df['Anno'], 'month': df['Mese'], 'day': 1})
    df.set_index('Data', inplace=True)
    df = df.sort_index().asfreq('MS').dropna()

    return df[['Valore venduto']]
```

2. Test della stazionarietà (`test_stationarity()`)

SARIMA richiede che la serie temporale sia stazionaria. Questa funzione utilizza il test di **Augmented Dickey-Fuller (ADF)** per verificare la stazionarietà:

- Se `p-value < 0.05`, la serie è stazionaria.
- Altrimenti, la serie va differenziata.

```
def test_stationarity(timeseries):
    result = adfuller(timeseries)
    print(f"ADF Statistic: {result[0]:.3f}, p-value: {result[1]:.3f}")
    if result[1] < 0.05:
        print("Serie stazionaria")
    else:
        print("Serie non stazionaria, necessaria differenziazione")
```

3. Identificazione dei parametri migliori (`find_best_sarima_order()`)

Questa funzione trova automaticamente la miglior combinazione di parametri SARIMA minimizzando il valore AIC (Akaike Information Criterion):

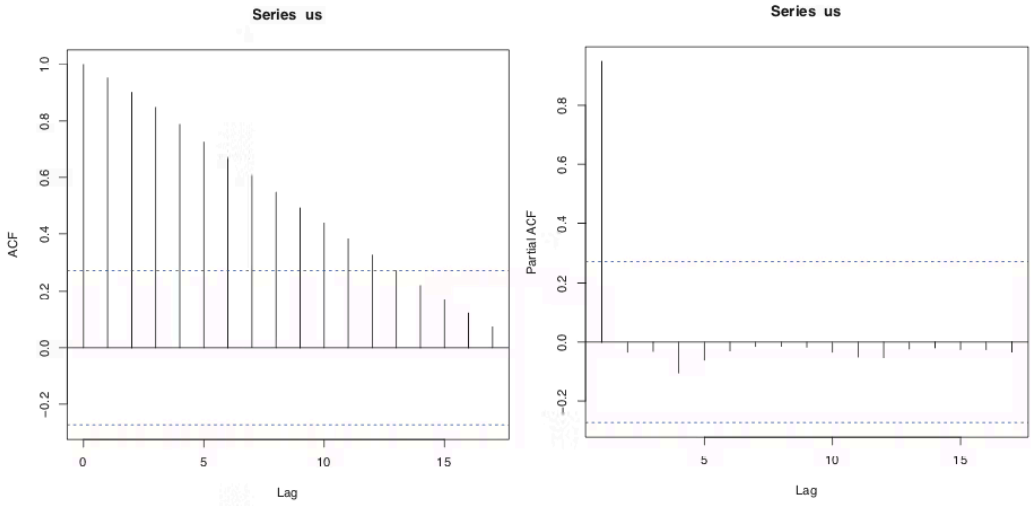
- Ciclo iterativo sui parametri (p,q,P,Q).
- Differenziazione stagionale (D=1) fissata perché tipicamente necessaria in vendite mensili.
- Ritorna i migliori parametri trovati.

```
def find_best_sarima_order(df):
    best_aic = float("inf")
    best_order = None
    best_seasonal_order = None
    d = 1 if adfuller(df['Valore venduto'])[1] > 0.05 else 0
    D = 1

    for p in range(3):
        for q in range(3):
            for P in range(2):
                for Q in range(2):
                    try:
                        model = SARIMAX(df, order=(p, d, q), seasonal_order=(P, D, Q, 12),
                                        enforce_stationarity=False,
enforce_invertibility=False)
                        model_fit = model.fit(disp=False)
                        if model_fit.aic < best_aic:
                            best_aic = model_fit.aic
                            best_order = (p, d, q)
                            best_seasonal_order = (P, D, Q, 12)
                    except:
                        continue
    return best_order, best_seasonal_order, best_aic
```

3.a Un altro modo per la scelta dei parametri ottimali

La scelta dei parametri del modello SARIMA può essere guidata dall'analisi dei grafici dell'**Autocorrelation Function (ACF)** e della **Partial Autocorrelation Function (PACF)**. L'**ACF** misura la correlazione tra il valore della serie temporale e i suoi lag passati, aiutando a identificare il numero di termini di media mobile (**q** per la parte non stagionale e **Q** per la parte stagionale). Se l'ACF si interrompe bruscamente dopo un certo lag, suggerisce un ordine q per il modello MA (Moving Average). La **PACF**, invece, mostra la correlazione diretta tra un valore e i suoi lag escludendo l'effetto dei lag intermedi, ed è utile per determinare il numero di termini autoregressivi (**p** per la parte non stagionale e **P** per la parte stagionale). Se la PACF si tronca bruscamente dopo un certo lag, indica un buon valore di **p**. Per identificare la componente stagionale, si osservano i picchi nei grafici ACF e PACF in corrispondenza del lag stagionale (ad esempio, ogni 12 mesi per dati mensili), il che aiuta a determinare i parametri **P** e **Q**. Questo metodo empirico viene spesso combinato con un'ottimizzazione automatica basata su criteri statistici come l'AIC o il BIC per trovare la combinazione ottimale di parametri.



(a) ACF (b) PACF

4. Addestramento e previsione (`train_sarima()`)

Addestra il modello SARIMA con i parametri trovati e genera previsioni future:

- Addestra solo con dati fino alla data indicata dall'utente.
- Fornisce una previsione e l'intervallo di confidenza associato.

```
def train_sarima(df, start_date, forecast_steps=12, order=(1,1,1), seasonal_order=(1,1,1,12)):
    df_filtered = df[df.index <= start_date]
    model = SARIMAX(df_filtered, order=order, seasonal_order=seasonal_order,
                    enforce_stationarity=False, enforce_invertibility=False)
    model_fit = model.fit()

    forecast_res = model_fit.get_forecast(steps=forecast_steps)
    forecast = forecast_res.predicted_mean
    conf_int = forecast_res.conf_int()

    return forecast, forecast.index, conf_int, model_fit.aic, model_fit.bic
```

5. Funzione principale (`main()`)

- Carica e analizza i dati.
- Richiede interattivamente all'utente:
    - Data di partenza previsione
    - Numero di mesi da prevedere
- Stampa grafico con dati storici, previsione e intervalli di confidenza.
- Mostra risultati numerici e metriche di valutazione (AIC e BIC).

Conclusione

Utilizzare il modello SARIMA, come mostrato dal codice, permette di generare previsioni accurate e affidabili, utili alla tua azienda per pianificare il budget stagionale, aiutare le buyer nelle scelte strategiche e ottimizzare l'intero ciclo di gestione degli ordini e della logistica. La struttura Python fornita è ideale perché automatizza tutte le fasi necessarie, dal pre-processing alla previsione finale, consentendo una gestione agile e precisa dei dati.

Previsione eseguita con i dati storici mensili dal 2016 ad oggi, con un modello Sarima(2,1,2)(1,1,1)12 confrontati con i dati reali di tutto il 2024.