

```
In [2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pmdarima import auto_arima
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
In [3]: df = pd.read_csv(r"C:\Users\HP\OneDrive\Desktop\NLP\Data\ML471_S4_Datafile_Conce

df['Datetime'] = pd.to_datetime(df['Datetime'])
df.set_index('Datetime', inplace=True)
```

```
In [4]: series = df['Consumption']

series = series.dropna()

train_size = int(len(series) * 0.8)
train = series[:train_size]
test = series[train_size:]

model = auto_arima(
    train,
    seasonal=True,
    m=12,
    trace=True,
    suppress_warnings=True,
    stepwise=True
)

print(model.summary())
```

Performing stepwise search to minimize aic

```
ARIMA(2,0,2)(1,1,1)[12] intercept : AIC=1363.439, Time=4.70 sec
ARIMA(0,0,0)(0,1,0)[12] intercept : AIC=1561.800, Time=0.05 sec
ARIMA(1,0,0)(1,1,0)[12] intercept : AIC=1413.742, Time=0.44 sec
ARIMA(0,0,1)(0,1,1)[12] intercept : AIC=1389.558, Time=0.31 sec
ARIMA(0,0,0)(0,1,0)[12] : AIC=1630.770, Time=0.04 sec
ARIMA(2,0,2)(0,1,1)[12] intercept : AIC=1361.464, Time=2.87 sec
ARIMA(2,0,2)(0,1,0)[12] intercept : AIC=1460.445, Time=0.33 sec
ARIMA(2,0,2)(0,1,2)[12] intercept : AIC=1363.425, Time=4.86 sec
ARIMA(2,0,2)(1,1,0)[12] intercept : AIC=1410.788, Time=1.98 sec
ARIMA(2,0,2)(1,1,2)[12] intercept : AIC=1364.708, Time=5.55 sec
ARIMA(1,0,2)(0,1,1)[12] intercept : AIC=1360.401, Time=1.19 sec
ARIMA(1,0,2)(0,1,0)[12] intercept : AIC=1460.030, Time=0.71 sec
ARIMA(1,0,2)(1,1,1)[12] intercept : AIC=1362.315, Time=2.75 sec
ARIMA(1,0,2)(0,1,2)[12] intercept : AIC=1362.275, Time=4.92 sec
ARIMA(1,0,2)(1,1,0)[12] intercept : AIC=1410.540, Time=1.80 sec
ARIMA(1,0,2)(1,1,2)[12] intercept : AIC=1363.088, Time=5.58 sec
ARIMA(0,0,2)(0,1,1)[12] intercept : AIC=1381.449, Time=0.91 sec
ARIMA(1,0,1)(0,1,1)[12] intercept : AIC=1369.107, Time=0.95 sec
ARIMA(1,0,3)(0,1,1)[12] intercept : AIC=1361.906, Time=4.03 sec
ARIMA(0,0,3)(0,1,1)[12] intercept : AIC=1372.386, Time=0.70 sec
ARIMA(2,0,1)(0,1,1)[12] intercept : AIC=1368.786, Time=1.72 sec
ARIMA(2,0,3)(0,1,1)[12] intercept : AIC=1363.645, Time=5.05 sec
ARIMA(1,0,2)(0,1,1)[12] : AIC=1364.129, Time=1.46 sec
```

Best model: ARIMA(1,0,2)(0,1,1)[12] intercept

Total fit time: 52.982 seconds

SARIMAX Results

=====

Dep. Variable: y No. Observations: 317

Model: SARIMAX(1, 0, 2)x(0, 1, [1], 12) Log Likelihood -674.201

Date: Sat, 31 Jan 2026 AIC 1360.401

Time: 09:11:49 BIC 1382.723

Sample: 01-01-1988 HQIC 1369.329

- 05-01-2014

Covariance Type: opg

```
=====
              coef    std err          z      P>|z|      [0.025      0.975]
-----
intercept    0.1107     0.065     1.694     0.090    -0.017     0.239
ar.L1        0.9302     0.041    22.451     0.000     0.849     1.011
ma.L1       -0.3395     0.079    -4.320     0.000    -0.494    -0.185
ma.L2       -0.3424     0.067    -5.113     0.000    -0.474    -0.211
ma.S.L12    -0.6950     0.048   -14.535     0.000    -0.789    -0.601
sigma2       4.7445     0.286    16.580     0.000     4.184     5.305
=====
```

```
==
Ljung-Box (L1) (Q):      0.05  Jarque-Bera (JB):      56.
92
Prob(Q):                0.83  Prob(JB):           0.
00
Heteroskedasticity (H):  2.48  Skew:              -0.
32
Prob(H) (two-sided):    0.00  Kurtosis:          5.
```

```
=====
==
```

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

```
In [5]: forecast = model.predict(n_periods=len(test))

mae = mean_absolute_error(test, forecast)
mape = np.mean(np.abs((test - forecast) / test)) * 100
rmse = np.sqrt(mean_squared_error(test, forecast))

print("MAE:", mae)
print("MAPE:", mape)
print("RMSE:", rmse)
```

MAE: 6.687000235923563

MAPE: 4.906826579404607

RMSE: 7.792387967823171

```
In [6]: plt.figure(figsize=(12,6))
plt.plot(train, label='Train', color='blue')
plt.plot(test, label='Actual', color='orange', linestyle='--')
plt.plot(test.index, forecast, label='Forecast', color='green', linestyle='--')

plt.title("SARIMA((1, 0, 2)x(0, 1, 1, 12) Forecast Plot")
plt.xlabel("Date")
plt.ylabel("Power Consumption")
plt.legend()
plt.grid(True)
plt.show()
```

