ARIMA 모델 (Autoregressive Integrated Moving Average)

- 시계열 예측에 가장 널리 사용되는 통계 모델 중 하나
- 시계열 데이터의 과거 패턴을 분석하여 미래를 예측
- 특히 데이터가 정상성(Stationarity)을 만족하지 않을 때 차분(Differencing)을 통해 정상 시계열로 변환한 후 모델링하는 특징을 가짐

ARIMA 모델의 구성 요소

- **AR (Autoregressive, 자기회귀)**: 이전 시점의 관측값($Y_{t-1}, Y_{t-2}, ...$)이 현재 시점의 관측값(Y_t)에 미치는 영향을 모델링합니다. 차수 `p`로 표현됩니다.
- **I (Integrated, 차분)**: 시계열을 정상 상태로 만들기 위해 필요한 차분 횟수를 나타냅니다. 차수 `d`로 표현됩니다.
- **MA (Moving Average, 이동평균)**: 이전 시점의 예측 오차($\epsilon_{t-1}, \epsilon_{t-2}, ...$)가 현재 시점의 관측값($Y_t$)에 미치는 영향을 모델링합니다. 차수 `q`로 표현됩니다.
- ARIMA 모델은 (p, d, q) 세 개의 차수로 표현됩니다.

관련 모델

- **AR(p) 모델**: 자기회귀 모델. $Y_t = c + \phi_1 Y_{t-1} + ... + \phi_p Y_{t-p} + \epsilon_t$
- **MA(q) 모델**: 이동평균 모델. $Y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q}$
- **ARMA(p, q) 모델**: AR과 MA를 결합한 모델. $Y_t = c + \phi_1 Y_{t-1} + ... + \phi_p Y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q}$
  - ARMA 모델은 데이터가 이미 정상성을 만족할 때 사용됩니다.
- **SARIMA(p, d, q)(P, D, Q)s 모델**: 계절성 ARIMA 모델. ARIMA 모델에 계절성(Seasonal) 요소를 추가한 모델입니다.
  - (p, d, q): 비계절성(non-seasonal) 부분의 차수.
  - (P, D, Q): 계절성(seasonal) 부분의 차수.
  - s: 계절성 주기(e.g., 월별 데이터면 12, 분기별이면 4).

적용 가능한 상황

- 시계열 데이터의 과거 패턴을 기반으로 미래 값을 예측해야 할 때.
- 데이터에 추세, 계절성, 자기상관 등의 패턴이 명확하게 존재할 때.
- 단기 및 중기 예측에 주로 사용됩니다.

구현 방법

`statsmodels` 라이브러리의 `ARIMA` 또는 `SARIMAX` 클래스를 사용합니다.

주의사항

- **정상성 확인**:
  - ARIMA 모델을 적용하기 전에 데이터의 정상성 여부를 반드시 확인해야 합니다.
  - 비정상 시계열인 경우 차분(d)을 통해 정상 시계열로 변환해야 합니다.

- **차수(p, d, q) 결정**:
  - ACF(자기상관 함수)와 PACF(편자기상관 함수) 플롯을 통해 AR(p)과 MA(q)의 차수를 결정하는 것이 일반적입니다.
  - ACF가 천천히 감소하고 PACF가 특정 시차에서 절단되면 AR 모델을,
  - ACF가 특정 시차에서 절단되고 PACF가 천천히 감소하면 MA 모델을,
  - 둘 다 천천히 감소하면 ARMA 모델을 고려합니다.
- **계절성 차수(P, D, Q) 결정**: 계절성 ACF/PACF 플롯을 통해 계절성 차수를 결정합니다.
- **모델 선택 기준**: 여러 ARIMA/SARIMA 모델을 비교할 때는 AIC(Akaike Information Criterion)나 BIC(Bayesian Information Criterion)와 같은 정보 기준을 사용합니다. 이 값들이 낮을수록 더 좋은 모델로 평가됩니다.

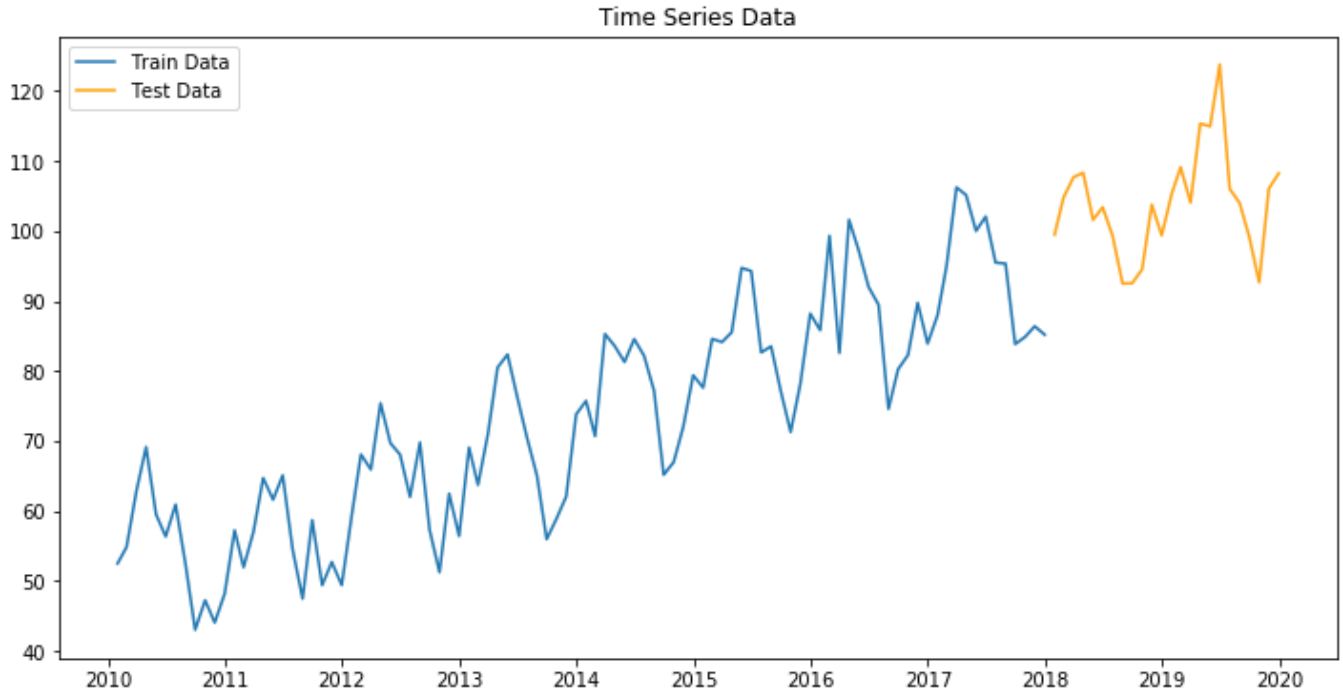
예제 데이터

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# 시계열 데이터 생성 (예시: 추세, 계절성, 노이즈 포함)
np.random.seed(42)
dates = pd.date_range(start='2010-01-01', periods=120, freq='M') # 10년치 월별 데이터
data = 50 + np.arange(120) * 0.5 + 10 * np.sin(np.arange(120) * 2 * np.pi / 12) + np.random.randn(120) * 5
ts = pd.Series(data, index=dates)

# 훈련/테스트 데이터 분할
train_size = int(len(ts) * 0.8)
train_data, test_data = ts[0:train_size], ts[train_size:]

plt.figure(figsize=(12, 6))
plt.plot(train_data, label='Train Data')
plt.plot(test_data, label='Test Data', color='orange')
plt.title('Time Series Data')
plt.legend()
plt.show()
```



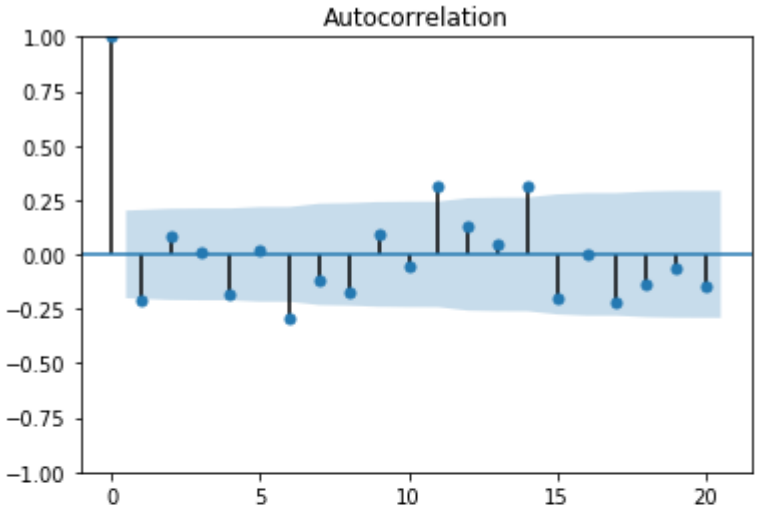
ARIMA 검정 및 모델

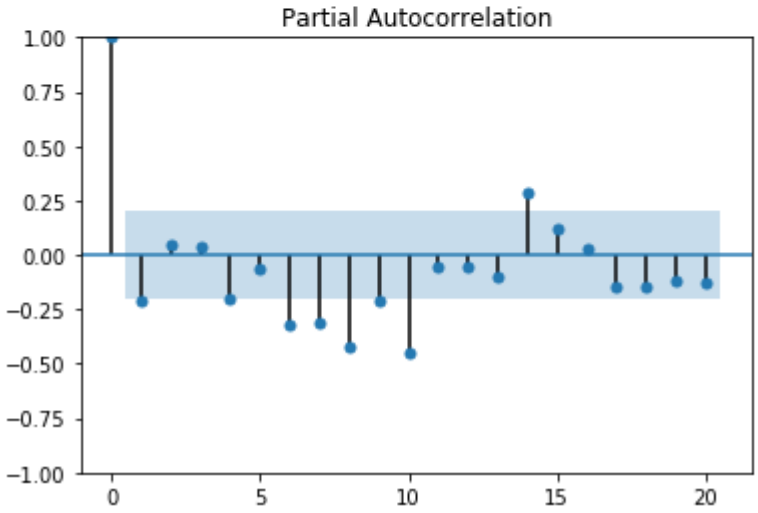
```
# ADF 검정 함수
def adf_test(series):
    result = adfuller(series, autolag='AIC')
    print('ADF Statistic: %f' % result[0])
    print('p-value: %f' % result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))

    if result[1] <= 0.05:
        print("귀무가설 기각 (p-value <= 0.05): 시계열은 정상성을 만족합니다.")
    else:
        print("귀무가설 채택 (p-value > 0.05): 시계열은 비정상 시계열입니다.")
# 차분 (d) 결정: ADF 검정 또는 시각적 확인
# (이 예시에서는 추세가 있으므로 1차 차분 필요)
ts_diff = train_data.diff().dropna()
adf_test(ts_diff) # 정상성 검정 수행
'''
ADF Statistic: -6.322415
p-value: 0.000000
Critical Values:
    1%: -3.513
    5%: -2.897
    10%: -2.586
귀무가설 기각 (p-value <= 0.05): 시계열은 정상성을 만족합니다.
'''
# AR(p) 및 MA(q) 차수 결정: ACF, PACF 플롯
plot_acf(ts_diff, lags=20)
plot_pacf(ts_diff, lags=20)
# (플롯을 보고 p, q 값 결정. 여기서는 임의로 p=2, q=2로 가정)
```

```
# ARIMA 모델 학습
# order=(p, d, q)
# p: AR 차수, d: 차분 차수, q: MA 차수
arima_model = ARIMA(train_data, order=(2, 1, 2)) # d=1 (1차 차분)
arima_results = arima_model.fit()

print("--- ARIMA 모델 결과 ---")
print(arima_results.summary())
'''
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   96
Model:                 ARIMA(2, 1, 2)   Log Likelihood                -319.435
Date:                Sun, 12 Oct 2025   AIC                            648.870
Time:                        21:51:37   BIC                            661.639
Sample:                    01-31-2010   HQIC                           654.030
                         - 12-31-2017
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.2914      1.179      0.247      0.805      -2.019       2.602
ar.L2          0.2891      0.736      0.393      0.694      -1.153       1.731
ma.L1         -0.5848      1.212     -0.483      0.629      -2.960       1.790
ma.L2         -0.2809      1.059     -0.265      0.791      -2.357       1.796
sigma2        48.5805      8.671      5.603      0.000      31.585      65.576
===================================================================================
=
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
1.29
Prob(Q):                              0.97   Prob(JB):
0.53
Heteroskedasticity (H):               1.28   Skew:
0.03
Prob(H) (two-sided):                  0.48   Kurtosis:
2.43
===================================================================================
=
'''
```

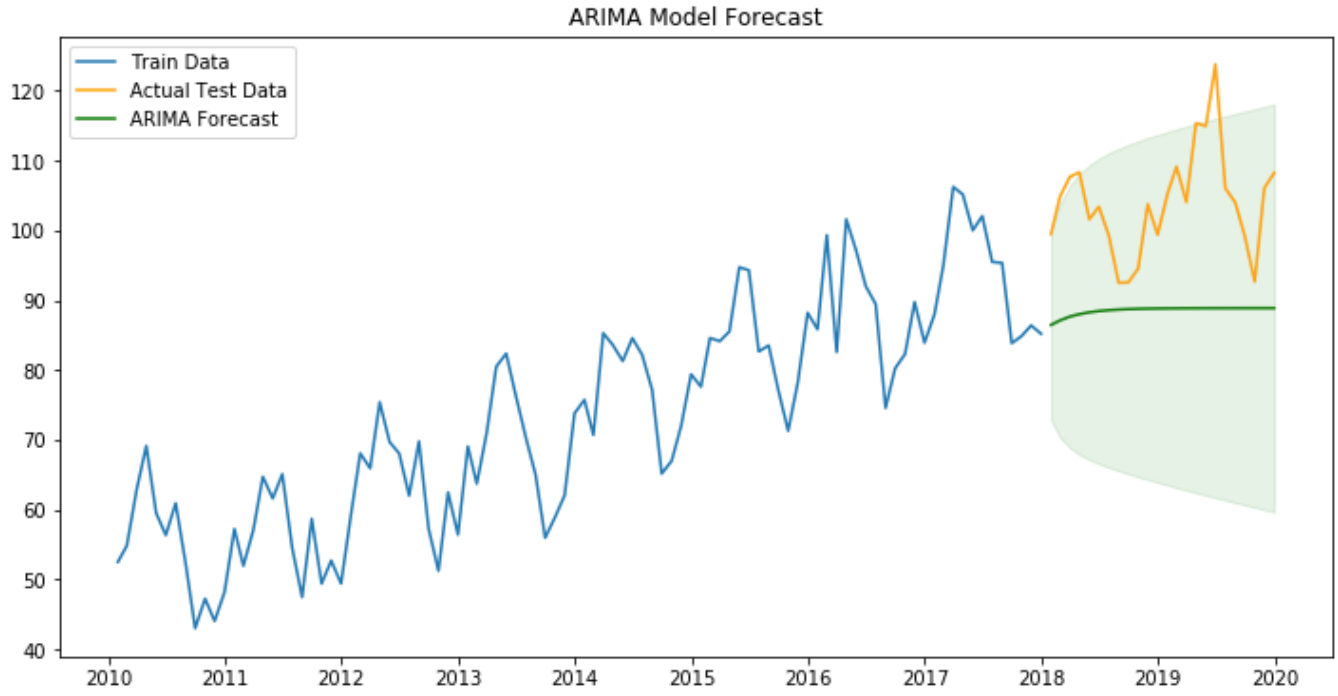




ARIMA 모델 예측

```
# 예측
# forecast(steps=예측 스텝 수)
# get_forecast(steps=예측 스텝 수).summary_frame()
forecast_steps = len(test_data)
forecast_result = arima_results.get_forecast(steps=forecast_steps)
forecast_mean = forecast_result.predicted_mean
conf_int = forecast_result.conf_int() # 신뢰 구간

# 결과 시각화
plt.figure(figsize=(12, 6))
plt.plot(train_data, label='Train Data')
plt.plot(test_data, label='Actual Test Data', color='orange')
plt.plot(forecast_mean, label='ARIMA Forecast', color='green')
plt.fill_between(conf_int.index, conf_int.iloc[:, 0], conf_int.iloc[:, 1],
color='green', alpha=0.1)
plt.title('ARIMA Model Forecast')
plt.legend()
plt.show()
```



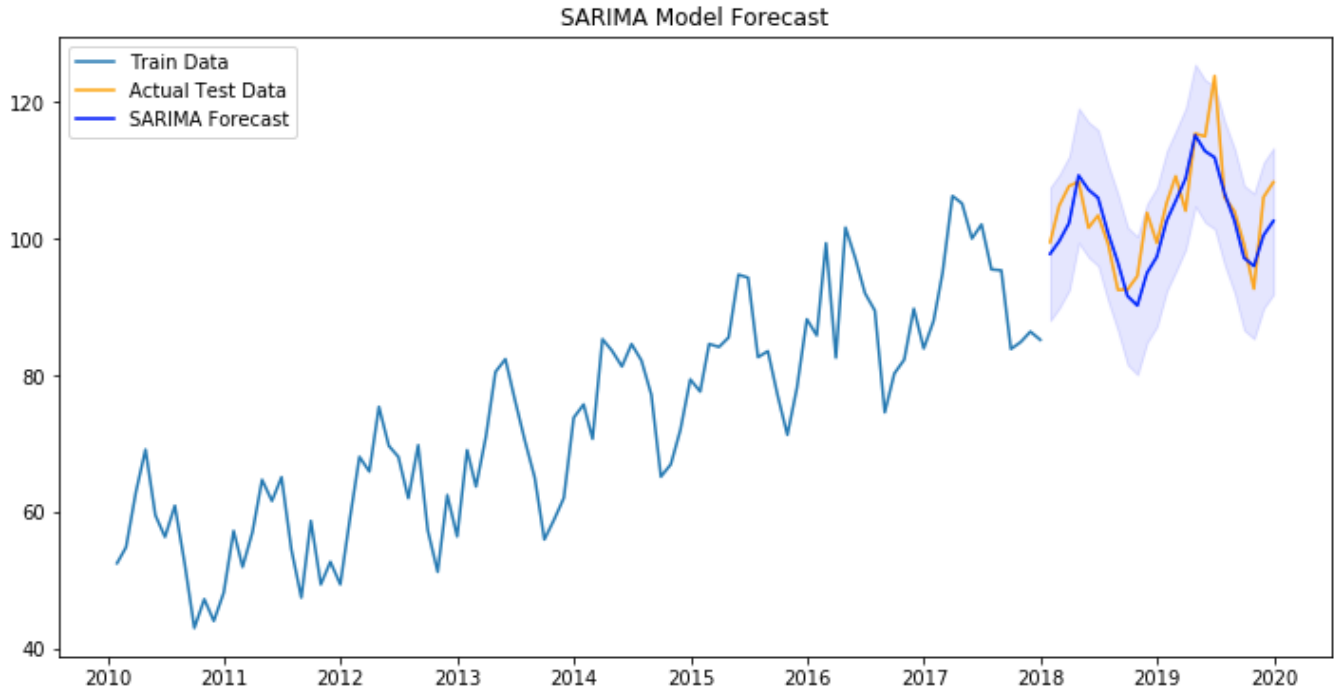

```
# SARIMA 모델 (계절성 포함 시계열)
# seasonal_order=(P, D, Q, s)
# P: 계절성 AR 차수, D: 계절성 차분 차수, Q: 계절성 MA 차수, s: 계절성 주기
sarima_model = ARIMA(train_data, order=(1, 1, 1), seasonal_order=(1, 1, 1, 12)) #
월별 계절성 주기 12
sarima_results = sarima_model.fit()

print("\n--- SARIMA 모델 결과 ---")
print(sarima_results.summary())
'''
                                     SARIMAX Results
==========================================================================================
======
Dep. Variable:                                   y   No. Observations:
96
Model:             ARIMA(1, 1, 1)x(1, 1, 1, 12)   Log Likelihood
-260.213
Date:                          Sun, 12 Oct 2025   AIC
530.426
Time:                                  21:58:21   BIC
542.520
Sample:                              01-31-2010   HQIC
535.285
                                   - 12-31-2017
Covariance Type:                             opg
==========================================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
ar.L1         -0.0901      0.159     -0.568      0.570      -0.401       0.221
ma.L1         -0.8969      0.097     -9.293      0.000      -1.086      -0.708
ar.S.L12      -0.0809      0.176     -0.459      0.646      -0.426       0.264
ma.S.L12      -0.9997    166.075     -0.006      0.995    -326.500     324.500
sigma2        21.8584   3628.128      0.006      0.995   -7089.142    7132.858
```

```
==========================================================================================
=
Ljung-Box (L1) (Q):                   0.18   Jarque-Bera (JB):
0.30
Prob(Q):                              0.67   Prob(JB):
0.86
Heteroskedasticity (H):               1.19   Skew:
0.10
Prob(H) (two-sided):                  0.65   Kurtosis:
3.22
==========================================================================================
=
'''

# 예측
sarima_forecast_result = sarima_results.get_forecast(steps=forecast_steps)
sarima_forecast_mean = sarima_forecast_result.predicted_mean
sarima_conf_int = sarima_forecast_result.conf_int()

plt.figure(figsize=(12, 6))
plt.plot(train_data, label='Train Data')
plt.plot(test_data, label='Actual Test Data', color='orange')
plt.plot(sarima_forecast_mean, label='SARIMA Forecast', color='blue')
plt.fill_between(sarima_conf_int.index, sarima_conf_int.iloc[:, 0],
sarima_conf_int.iloc[:, 1], color='blue', alpha=0.1)
plt.title('SARIMA Model Forecast')
plt.legend()
plt.show()
```



결과 해석 방법

- `summary()`: 모델의 통계적 유의성, 계수, AIC, BIC 등을 포함한 상세한 결과를 제공합니다.

    - `coef`: 각 AR, MA 계수 및 상수항의 추정치.
    - `P>|z|`: 각 계수의 p-value. 0.05보다 작으면 통계적으로 유의미하다고 판단.
    - `AIC`, `BIC`: 모델의 적합도와 복잡도를 동시에 고려하는 지표. 값이 작을수록 좋은 모델.
- **예측 플롯**: 실제 테스트 데이터와 모델의 예측값을 비교하여 시각적으로 모델의 성능을 평가합니다. 신뢰 구간(Confidence Interval)을 함께 표시하여 예측의 불확실성을 파악할 수 있습니다.

장단점 및 대안

- **장점**:
    - 시계열 데이터의 추세, 계절성, 자기상관을 효과적으로 모델링할 수 있습니다.
    - 통계적 기반이 탄탄하여 모델의 해석이 용이합니다.
    - 단기 예측에서 높은 정확도를 보입니다.
- **단점**:
    - 데이터가 정상성을 만족해야 하므로, 비정상 시계열인 경우 차분 등의 전처리 과정이 필요합니다.
    - 모델의 차수(p, d, q)를 결정하는 과정이 다소 주관적이고 경험에 의존할 수 있습니다.
    - 장기 예측에는 적합하지 않을 수 있으며, 갑작스러운 변화나 이벤트에 취약합니다.
- **대안**:
    - **Prophet**: 휴일 효과나 여러 계절성을 자동으로 처리하며, 비전문가도 쉽게 사용할 수 있는 시계열 예측 라이브러리입니다.
    - **VAR (Vector Autoregression)**: 여러 시계열 간의 상호작용을 모델링하는 데 사용됩니다.
    - **머신러닝/딥러닝 모델**: 시계열 데이터를 특성으로 변환하여 일반적인 회귀 모델을 적용하거나, LSTM, GRU와 같은 순환 신경망(RNN)을 사용하여 복잡한 시계열 패턴을 학습할 수 있습니다.
- SARIMA 모델은 `statsmodels.tsa.statespace.sarimax.SARIMAX` 을 통해서도 구현 가능
- MARIMA: Multivariate ARIMA 모델이 존재, `statsmodels.tsa.statespace.varmax.VARMAX` 을 통해서 구현 가능