고급 시계열 모델 (Advanced Time Series Models)

전통적인 ARIMA 계열 모델 외에도, 시계열 데이터의 복잡한 특성(다변량 관계, 비선형성, 다양한 계절성 등)을 더 효과적으로 모델링하고 예측하기 위한 다양한 고급 시계열 모델들이 존재합니다. 이러한 모델들은 특정 상황이나 데이터 유형에 더 적합하며, 때로는 더 높은 예측 성능을 제공합니다.

주의사항

- **데이터 전처리**: 각 모델의 특성에 맞는 데이터 전처리(스케일링, 결측치 처리 등)가 필요합니다.
- **하이퍼파라미터 튜닝**: 대부분의 고급 모델은 튜닝해야 할 하이퍼파라미터가 많으며, 최적의 성능을 위해 교차 검증 등을 통한 신중한 튜닝이 필요합니다.
- **계산 비용**: 특히 LSTM과 같은 딥러닝 모델은 학습에 상당한 시간과 컴퓨팅 자원을 요구합니다.

VAR (Vector Autoregression, 벡터 자기회귀)

두 개 이상의 상호 연관된 시계열 데이터의 관계를 모델링하고 예측하는 데 사용되는 모델입니다. 각 시계열이 자기 자신의 과거 값뿐만 아니라, 다른 시계열들의 과거 값에도 영향을 받는다고 가정합니다.

- **적용**: 거시 경제 지표(GDP, 물가상승률, 금리 등)와 같이 여러 변수가 서로 영향을 주고받는 시스템을 분석하고 예측할 때 유용합니다.
- **장점**: 여러 시계열 간의 동적인 관계를 동시에 모델링할 수 있습니다.
- **단점**: 모델의 차수(lag)가 증가하면 추정해야 할 파라미터 수가 기하급수적으로 늘어나 계산 비용이 커지고 과적합 위험이 있습니다.
- **구현 방법**: `statsmodels` 라이브러리의 `VAR` 클래스.

```
import pandas as pd
import numpy as np
from statsmodels.tsa.api import VAR
from statsmodels.tsa.stattools import adfuller
import matplotlib.pyplot as plt

# 1. 데이터 생성 (두 개의 상호 연관된 시계열)
np.random.seed(42)
data = np.zeros((100, 2))
for i in range(1, 100):
    data[i, 0] = 0.5 * data[i-1, 0] + 0.2 * data[i-1, 1] + np.random.randn()
    data[i, 1] = 0.3 * data[i-1, 0] + 0.6 * data[i-1, 1] + np.random.randn()
df_var = pd.DataFrame(data, columns=['series1', 'series2'])

# 2. 정상성 확인 (ADF 검정)
# VAR 모델은 모든 시계열이 정상성을 만족해야 함.
# 비정상 시계열인 경우 차분 필요.
print("--- VAR 모델을 위한 정상성 검정 ---")
for name, series in df_var.items():
    result = adfuller(series)
    print(f'{name}: p-value = {result[1]:.3f}')
```

```
    if result[1] > 0.05:
        print(f'{name}은 비정상 시계열입니다. 차분이 필요할 수 있습니다.')
# series1: p-value = 0.000
# series2: p-value = 0.000

# 3. VAR 모델 학습
# maxlags: 고려할 최대 시차.
model_var = VAR(df_var)
results_var = model_var.fit(maxlags=5, ic='aic') # ic='aic'로 최적 시차 자동 선택
print("\n--- VAR 모델 결과 ---")
print(results_var.summary())
'''
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Sun, 12, Oct, 2025
Time:                     22:09:19
--------------------------------------------------------------------
No. of Equations:         2.00000    BIC:                   0.0426393
Nobs:                     98.0000    HQIC:                  -0.114442
Log likelihood:          -257.276    FPE:                    0.801752
AIC:                    -0.221133    Det(Omega_mle):         0.725802
--------------------------------------------------------------------
Results for equation series1
================================================================================
                coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------
const             -0.154280         0.090785           -1.699           0.089
L1.series1         0.456790         0.102530            4.455           0.000
L1.series2         0.103499         0.087760            1.179           0.238
L2.series1        -0.086592         0.106874           -0.810           0.418
L2.series2         0.115433         0.084156            1.372           0.170
================================================================================

Results for equation series2
================================================================================
                coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------
const              0.100484         0.104064            0.966           0.334
L1.series1         0.267168         0.117527            2.273           0.023
L1.series2         0.534523         0.100596            5.314           0.000
L2.series1         0.285399         0.122506            2.330           0.020
L2.series2        -0.070034         0.096465           -0.726           0.468
================================================================================

Correlation matrix of residuals
             series1   series2
series1    1.000000  0.070124
series2    0.070124  1.000000
'''
# 4. 예측
lag_order = results_var.k_ar
forecast_input = df_var.values[-lag_order:]
```

```
forecast_var = results_var.forecast(y=forecast_input, steps=10)
forecast_df_var = pd.DataFrame(forecast_var,
index=pd.date_range(start=df_var.index[-1] + 1, periods=10), columns=
['series1_pred', 'series2_pred'])

print("\n--- VAR 예측 결과 ---")
print(forecast_df_var)
'''
                              series1_pred  series2_pred
1970-01-01 00:00:00.000000100     -0.626014     -0.165556
1970-01-02 00:00:00.000000100     -0.377418     -0.427989
1970-01-03 00:00:00.000000100     -0.335880     -0.396190
1970-01-04 00:00:00.000000100     -0.365434     -0.278766
1970-01-05 00:00:00.000000100     -0.366707     -0.214268
1970-01-06 00:00:00.000000100     -0.344499     -0.196791
1970-01-07 00:00:00.000000100     -0.324991     -0.186396
1970-01-08 00:00:00.000000100     -0.314909     -0.170514
1970-01-09 00:00:00.000000100     -0.309149     -0.154491
1970-01-10 00:00:00.000000100     -0.303900     -0.142623
'''
```

결과 해석

- `summary()`를 통해 각 시계열이 다른 시계열의 과거 값에 미치는 영향(계수)을 확인할 수 있습니다.
- 충격 반응 함수(Impulse Response Function)를 통해 한 변수의 충격이 다른 변수에 미치는 동적인 영향을 분석할 수 있습니다.

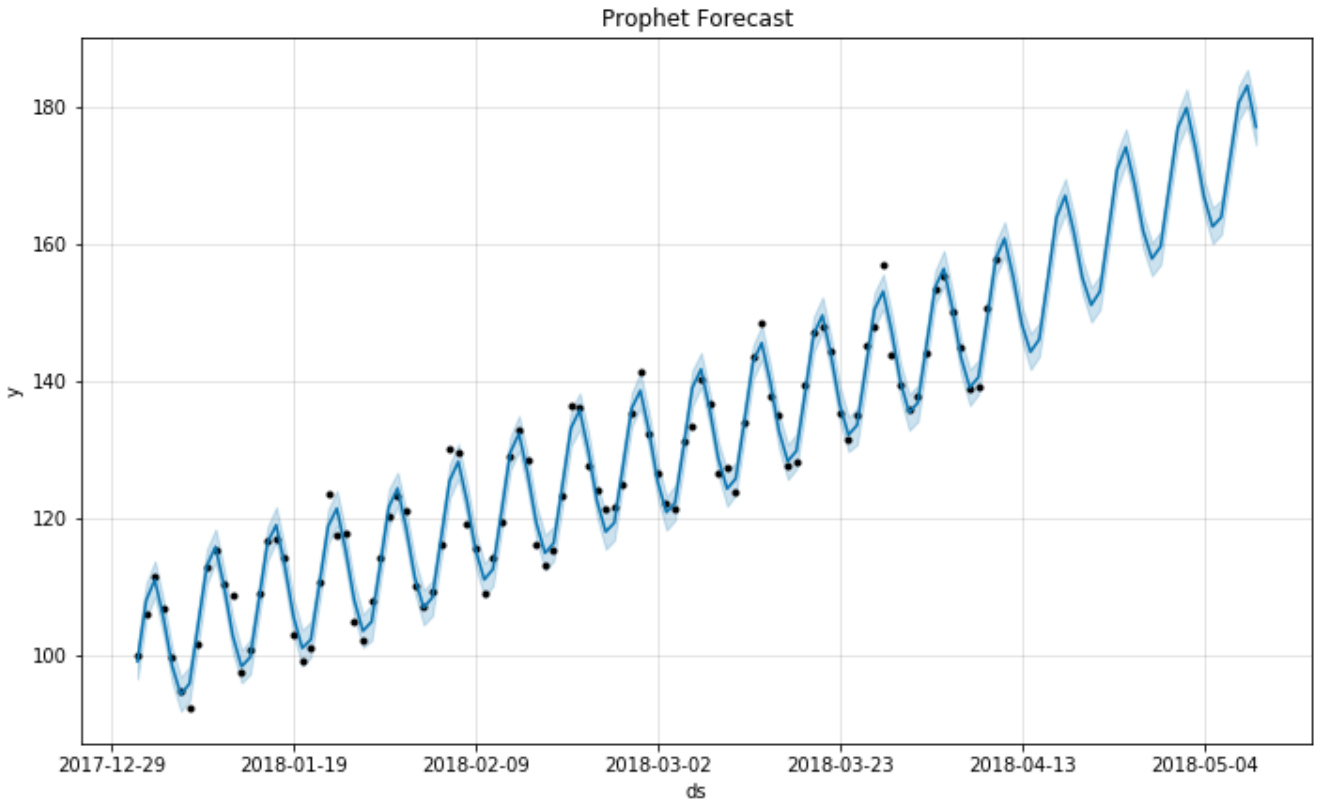
Prophet

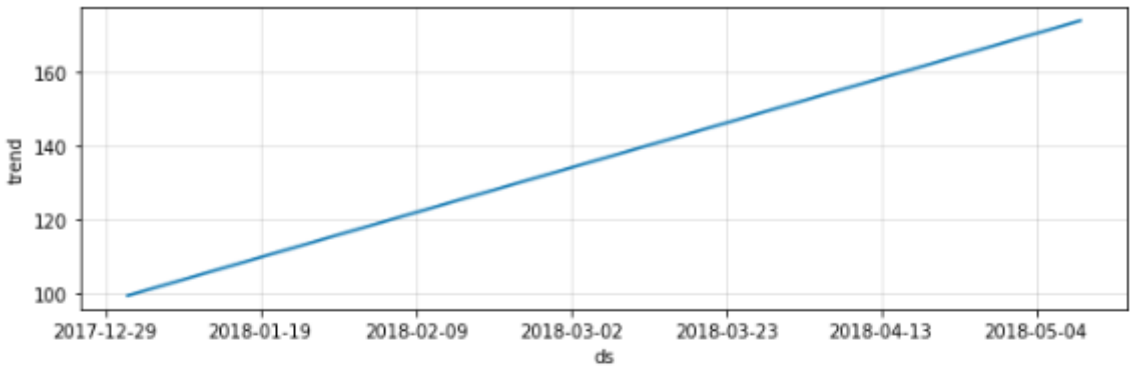
페이스북에서 개발한 시계열 예측 라이브러리로, 비전문가도 쉽게 사용할 수 있도록 설계되었습니다. 시계열 데이터를 추세(trend), 계절성(seasonality), 휴일(holidays)의 세 가지 주요 구성 요소로 분해하여 모델링합니다.

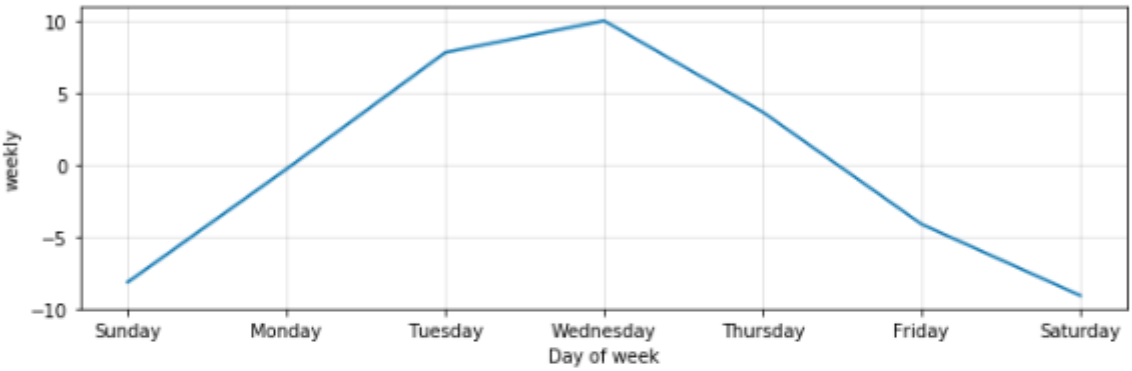
- **적용**: 비즈니스 예측(판매량, 웹 트래픽 등)과 같이 강한 계절성, 휴일 효과, 누락된 데이터, 이상치 등이 있는 시계열 데이터에 특히 효과적입니다.
- **장점**:
  - 직관적인 파라미터 설정으로 쉽게 모델을 구축하고 튜닝할 수 있습니다.
  - 여러 계절성(일별, 주별, 연별)과 휴일 효과를 자동으로 처리합니다.
  - 누락된 데이터나 이상치에 강건합니다.
- **단점**: ARIMA 모델처럼 시계열의 자기상관 구조를 명시적으로 모델링하지는 않습니다.
- **구현 방법**: `fbprophet` 라이브러리.

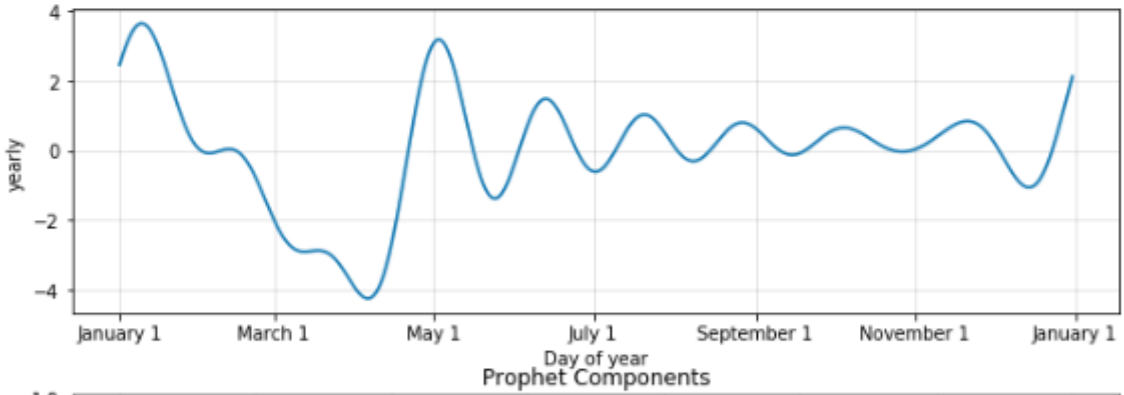
```
# !pip install prophet
from prophet import Prophet

# 1. 데이터 준비 (Prophet은 'ds'와 'y' 컬럼 필요)
df_prophet = pd.DataFrame({
    'ds': pd.to_datetime(pd.date_range(start='2018-01-01', periods=100,
```

```
freq='D')),
    'y': 100 + np.arange(100) * 0.5 + 10 * np.sin(np.arange(100) * 2 * np.pi / 7)
+ np.random.randn(100) * 2
})

# 2. Prophet 모델 학습
# seasonality_mode: 계절성 유형 ('additive' 또는 'multiplicative')
# daily_seasonality, weekly_seasonality, yearly_seasonality: 각 계절성 자동 감지 여
부
m = Prophet(daily_seasonality=True, weekly_seasonality=True,
yearly_seasonality=True)
m.fit(df_prophet)

# 3. 미래 데이터프레임 생성 및 예측
future = m.make_future_dataframe(periods=30) # 30일 미래 예측
forecast_prophet = m.predict(future)

# 4. 결과 시각화
fig = m.plot(forecast_prophet)
plt.title('Prophet Forecast')
plt.show()

fig2 = m.plot_components(forecast_prophet)
plt.title('Prophet Components')
plt.show()
```

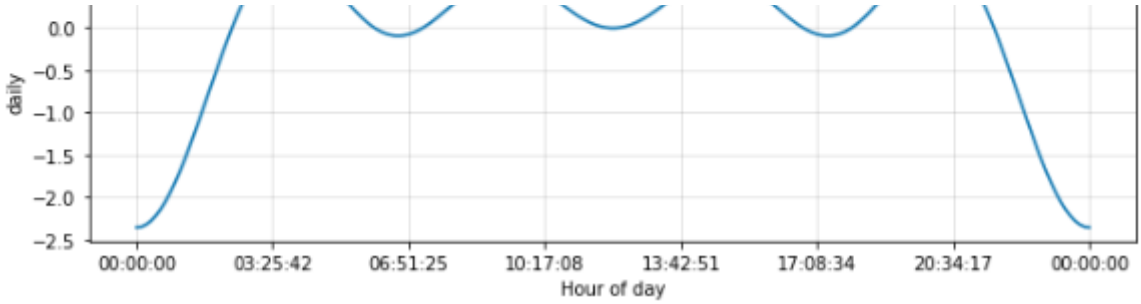

Prophet Forecast
y
ds
180
160
140
120
100
2017-12-29
2018-01-19
2018-02-09
2018-03-02
2018-03-23
2018-04-13
2018-05-04


trend
ds
160
140
120
100
2017-12-29
2018-01-19
2018-02-09
2018-03-02
2018-03-23
2018-04-13
2018-05-04


weekly
Day of week
10
5
0
−5
−10
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday


yearly
Day of year
4
2
0
−2
−4
January 1
March 1
May 1
July 1
September 1
November 1
January 1


Prophet Components
1.0
0.5



결과 해석

- `plot_components()`를 통해 추세, 주별 계절성, 연별 계절성 등 모델이 학습한 각 구성 요소를 시각적으로 확인할 수 있어 해석이 용이합니다.

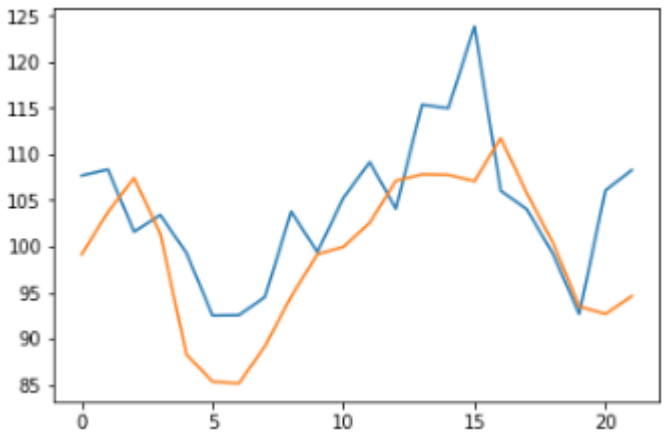
LSTM (Long Short-Term Memory, 장단기 기억 신경망)

순환 신경망(Recurrent Neural Network, RNN)의 한 종류로, 시계열과 같이 순차적인 데이터에서 장기적인 의존성(long-term dependencies)을 학습하는 데 특화된 딥러닝 모델입니다. '게이트(gate)' 메커니즘을 통해 정보를 기억하거나 잊어버리는 능력을 가집니다.

- **적용**: 주가 예측, 자연어 처리(번역, 텍스트 생성), 음성 인식 등 복잡하고 비선형적인 시계열 패턴을 학습해야 할 때 사용됩니다.
- **장점**: 복잡한 비선형 패턴과 장기적인 의존성을 효과적으로 학습할 수 있습니다.
- **단점**: 대량의 데이터와 높은 계산 자원이 필요하며, 모델의 해석이 매우 어렵습니다. 과적합 위험이 높고, 하이퍼파라미터 튜닝이 까다롭습니다.
- **구현 방법**: `TensorFlow` 또는 `PyTorch`와 같은 딥러닝 프레임워크.

```
# !pip install tensorflow
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler

# 1. 데이터 준비 (시계열 데이터를 지도 학습 형태로 변환)
# 예시: 이전 10일 데이터로 다음 1일 예측
def create_dataset(data, look_back=1):
    X, Y = [], []
    for i in range(len(data) - look_back):
        X.append(data[i:(i + look_back), 0])
        Y.append(data[i + look_back, 0])
    return np.array(X), np.array(Y)

# 데이터 스케일링
scaler_lstm = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler_lstm.fit_transform(ts.values.reshape(-1, 1))

look_back = 10
X_lstm, y_lstm = create_dataset(scaled_data, look_back)

# LSTM 입력 형태 (samples, time steps, features)
X_lstm = np.reshape(X_lstm, (X_lstm.shape[0], X_lstm.shape[1], 1))
```

```
# 훈련/테스트 분할
train_size_lstm = int(len(X_lstm) * 0.8)
X_train_lstm, X_test_lstm = X_lstm[0:train_size_lstm],
X_lstm[train_size_lstm:len(X_lstm)]
y_train_lstm, y_test_lstm = y_lstm[0:train_size_lstm],
y_lstm[train_size_lstm:len(y_lstm)]

# 2. LSTM 모델 구축
model_lstm = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(50, activation='relu', input_shape=(look_back, 1)),
    tf.keras.layers.Dense(1)
])
model_lstm.compile(optimizer='adam', loss='mse')

# 3. 모델 학습
model_lstm.fit(X_train_lstm, y_train_lstm, epochs=100, batch_size=1, verbose=0)

# 4. 예측
train_predict_lstm = model_lstm.predict(X_train_lstm)
test_predict_lstm = model_lstm.predict(X_test_lstm)

# 스케일 역변환
train_predict_lstm = scaler_lstm.inverse_transform(train_predict_lstm)
y_train_lstm_inv = scaler_lstm.inverse_transform(y_train_lstm.reshape(-1, 1))
test_predict_lstm = scaler_lstm.inverse_transform(test_predict_lstm)
y_test_lstm_inv = scaler_lstm.inverse_transform(y_test_lstm.reshape(-1, 1))

# 5. 결과 시각화 (생략, 실제 데이터와 예측값 비교)
plt.plot(y_test_lstm_inv, label='Actual')
plt.plot(test_predict_lstm, label='Predicted')
plt.show()
```



결과 해석

- 딥러닝 모델이므로 직접적인 해석은 어렵습니다. 주로 예측 성능(MSE, RMSE)과 시각화를 통해 실제값과 예측값의 일치도를 확인합니다.

상태 공간 모델 (State Space Models)

관측할 수 없는 '상태(state)' 변수를 도입하여 시계열의 동적인 변화를 모델링하는 통계적 프레임워크입니다. 칼만 필터(Kalman Filter)와 같은 알고리즘을 사용하여 상태 변수를 추정하고 예측합니다.

- **적용**: 시계열의 추세, 계절성, 회귀 효과 등을 통합적으로 모델링할 수 있으며, 누락된 데이터 처리나 실시간 예측에 유용합니다.
- **장점**: 유연하고 다양한 시계열 구성 요소를 통합적으로 다룰 수 있습니다.
- **단점**: 모델의 설정이 복잡하고, 해석이 어려울 수 있습니다.
- **구현 방법**: `statsmodels` 라이브러리의 `SARIMAX` (내부적으로 상태 공간 모델 기반) 또는 `UnobservedComponents` 클래스.

장단점 요약

- **VAR**:
  - **장점**: 여러 시계열 간의 상호작용을 모델링하여 더 포괄적인 분석이 가능합니다.
  - **단점**: 변수 수가 많아지면 모델 복잡도가 급증하고, 모든 시계열이 정상성을 만족해야 합니다.
- **Prophet**:
  - **장점**: 사용하기 쉽고, 강한 계절성 및 휴일 효과가 있는 비즈니스 시계열 데이터에 매우 효과적입니다.
  - **단점**: ARIMA 모델처럼 시계열의 자기상관 구조를 정교하게 모델링하지는 않습니다.
- **LSTM**:
  - **장점**: 복잡한 비선형 패턴과 장기적인 의존성을 학습하는 데 탁월합니다.
  - **단점**: 대량의 데이터와 컴퓨팅 자원이 필요하며, 모델의 해석이 어렵고 튜닝이 까다롭습니다.
- **상태 공간 모델**:
  - **장점**: 유연한 모델링 프레임워크를 제공하며, 누락된 데이터 처리나 실시간 예측에 강점을 가집니다.
  - **단점**: 모델 설정이 복잡하고, 통계적 배경 지식이 필요합니다.