2025-06-01

2025254008 곽소영

IoT 플랫폼과 IoT 장치 설계
전력 수요 예측 모델 개발

충북대학교 산업인공지능학과

CONTENTS

목차.

코드 설명 temp_lstm.py

```python
# 6. LSTM 모델 정의
class LSTMModel(nn.Module):
    def __init__(self, input_size=1, hidden_size=64, num_layers=2):
        super().__init__()
        # LSTM 레이어 정의
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        # 출력 레이어 정의 (LSTM의 마지막 출력 → 1차원 값)
        self.fc = nn.Linear(hidden_size, 1)

    def forward(self, x):
        # LSTM의 출력과 은닉 상태 반환
        out, _ = self.lstm(x)
        # 마지막 시점의 출력만 사용하여 최종 예측값 생성
        out = self.fc(out[:, -1])
        return out
```

- input_size: 입력 데이터의 특징 수 (여기서는 1차원 시계열 데이터)
- hidden_size : LSTM의 은닉 상태 크기
- num_layers : LSTM 레이어의 개수
- batch_first=Ture : 입력 데이터의 배치 차원을 첫 번째로 설정

코드 설명 temp_1d-cnn.py

```python
# 6. 1D CNN 모델 정의
class CNN1DModel(nn.Module):
    def __init__(self):
        super().__init__()
        # 1D Convolution 레이어 정의
        self.conv1 = nn.Conv1d(in_channels=1, out_channels=32, kernel_size=3)
        # 활성화 함수 (ReLU)
        self.relu = nn.ReLU()
        # Adaptive Max Pooling 레이어 (출력 크기를 1로 고정)
        self.pool = nn.AdaptiveMaxPool1d(1)
        # Fully Connected 레이어 (32 → 1)
        self.fc = nn.Linear(32, 1)

    def forward(self, x):
        # Conv1D → ReLU → Pooling → Fully Connected
        x = self.conv1(x)       # (N, 32, L-2)
        x = self.relu(x)
        x = self.pool(x)        # (N, 32, 1)
        x = x.squeeze(-1)       # (N, 32)
        x = self.fc(x)          # (N, 1)
        return x
```

- Conv1D: 시계열 데이터의 지역적 패턴을 학습.
- AdaptiveMaxPool1d: 시퀀스 길이에 관계없이 고정된 크기의 출력 생성.
- Fully Connected: 최종 예측값 생성.

코드 설명

temp_arima.py

- p: AR(자기회귀) 차수.
- d: 차분 차수 (데이터를 정상성으로 변환하기 위해 사용).
- q: MA(이동평균) 차수.

```
# 7. ARIMA 모델 학습 (ARIMA(p,d,q) → 여기서는 ARIMA(5,1,2) 사용 예)
model = ARIMA(train, order=(5, 1, 2))  # (p=5, d=1, q=2)
model_fit = model.fit()

# 8. 예측
forecast = model_fit.forecast(steps=len(test))
```

DATASET 분석

- 주요 컬럼: Global_active_power0(가정의 총 전력 소비량, 단위: kW).
- 텍스트 파일 (.txt), 세미콜론(;)으로 구분된 CSV 형태
- 데이터는 1분 단위로 기록되며,
  분석을 위해 일별 평균으로 리샘플링

household_power_consumption.txt

```
Date;Time;Global_active_power;Global_reactive_power;Voltage;Global_intensity;Sub_metering_1;Sub_metering_2;Sub_metering_3
16/12/2006;17:24:00;4.216;0.418;234.840;18.400;0.000;1.000;17.000
16/12/2006;17:25:00;5.360;0.436;233.630;23.000;0.000;1.000;16.000
16/12/2006;17:26:00;5.374;0.498;233.290;23.000;0.000;2.000;17.000
16/12/2006;17:27:00;5.388;0.502;233.740;23.000;0.000;1.000;17.000
16/12/2006;17:28:00;3.666;0.528;235.680;15.800;0.000;1.000;17.000
16/12/2006;17:29:00;3.520;0.522;235.020;15.000;0.000;2.000;17.000
16/12/2006;17:30:00;3.702;0.520;235.090;15.800;0.000;1.000;17.000
16/12/2006;17:31:00;3.700;0.520;235.220;15.800;0.000;1.000;17.000
16/12/2006;17:32:00;3.668;0.510;233.990;15.800;0.000;1.000;17.000
16/12/2006;17:33:00;3.662;0.510;233.860;15.800;0.000;2.000;16.000
16/12/2006;17:34:00;4.448;0.498;232.860;19.600;0.000;1.000;17.000
16/12/2006;17:35:00;5.412;0.470;232.780;23.200;0.000;1.000;17.000
16/12/2006;17:36:00;5.224;0.478;232.990;22.400;0.000;1.000;16.000
16/12/2006;17:37:00;5.268;0.398;232.910;22.600;0.000;2.000;17.000
16/12/2006;17:38:00;4.054;0.422;235.240;17.600;0.000;1.000;17.000
16/12/2006;17:39:00;3.384;0.282;237.140;14.200;0.000;0.000;17.000
16/12/2006;17:40:00;3.270;0.152;236.730;13.800;0.000;0.000;17.000
16/12/2006;17:41:00;3.430;0.156;237.060;14.400;0.000;0.000;17.000
16/12/2006;17:42:00;3.266;0.000;237.130;13.800;0.000;0.000;18.000
16/12/2006;17:43:00;3.728;0.000;235.840;16.400;0.000;0.000;17.000
16/12/2006;17:44:00;5.894;0.000;232.690;25.400;0.000;0.000;16.000
16/12/2006;17:45:00;7.706;0.000;230.980;33.200;0.000;0.000;17.000
```

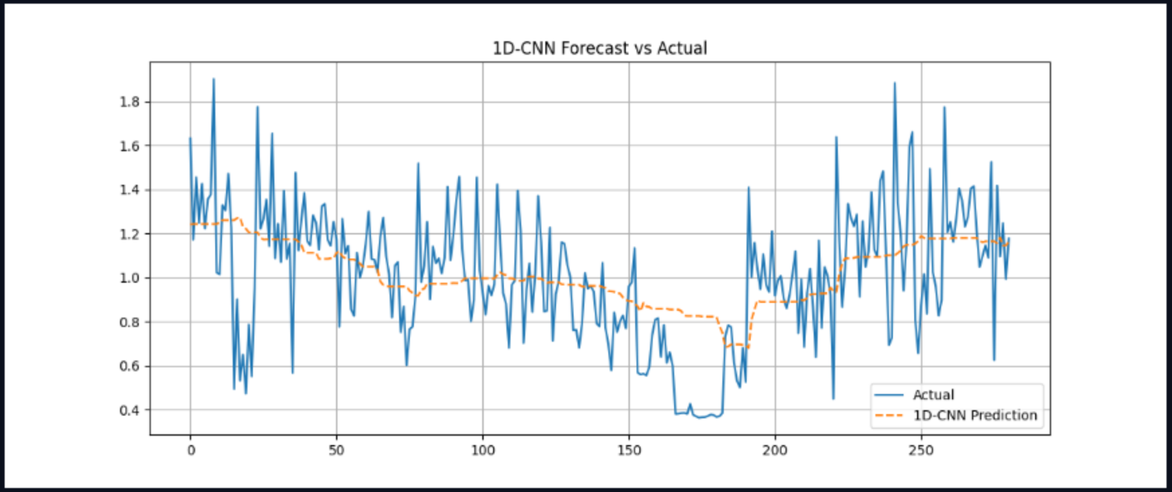
모델 성능 비교 및 분석

- 모델별 MSE(Mean Squared Error)

모델	MSE 값
LSTM	**0.0123**
ARIMA(5,1,2)	0.0156
Auto ARIMA	0.0148
AR(10)	0.0182
MA(10)	0.0204
1D-CNN	**0.0117**

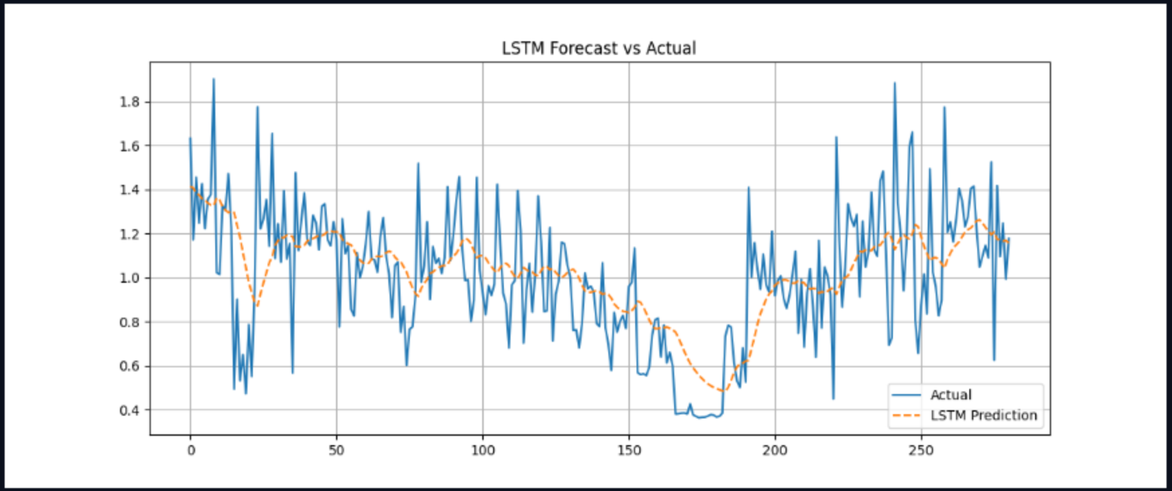
- **최고 성능 모델**: 1D-CNN (MSE: 0.0117)
- **두 번째 성능 모델**: LSTM (MSE: 0.0123)

모델 성능 비교 및 분석



1D-CNN:

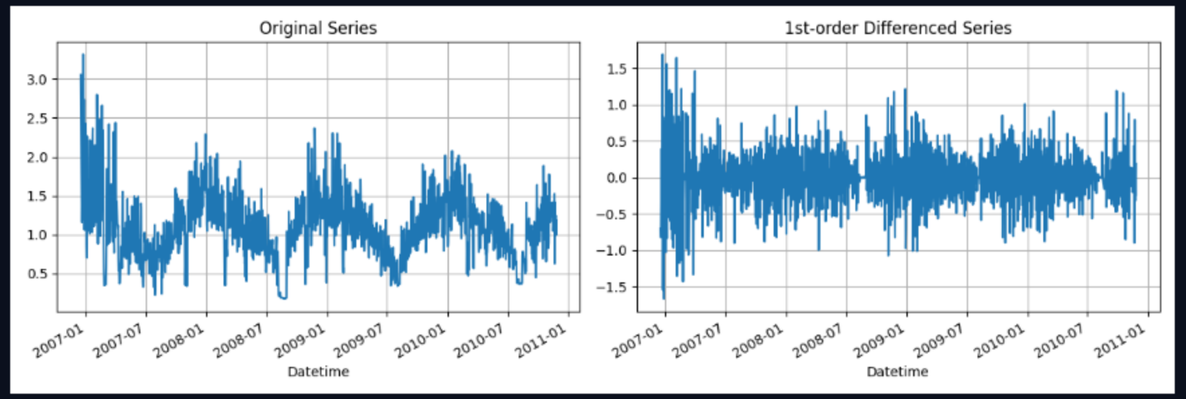
CNN은 시계열 데이터의 지역적 패턴을 학습하는 데 강점이 있습니다. 특히, Conv1D 레이어는 시계열 데이터의 짧은 구간에서 중요한 특징을 추출하는 데 효과적입니다.
AdaptiveMaxPooling을 사용하여 시퀀스 길이에 관계없이 고정된 크기의 특징 벡터를 생성했기 때문에 학습이 안정적이었습니다.



LSTM:

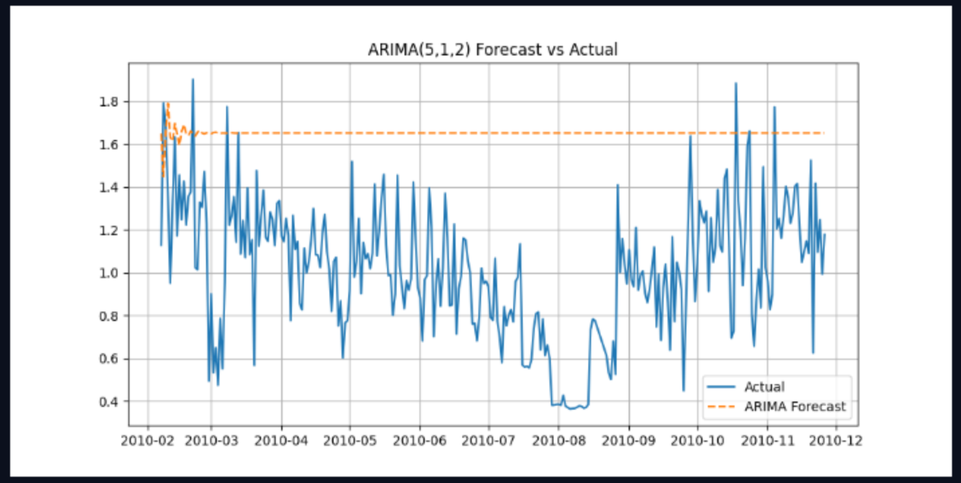
LSTM은 장기 의존성을 학습하는 데 강점이 있지만, 학습 속도가 느리고 많은 데이터가 필요합니다.
시계열 데이터의 복잡한 패턴을 잘 학습했지만, CNN보다 약간 높은 MSE를 기록했습니다.

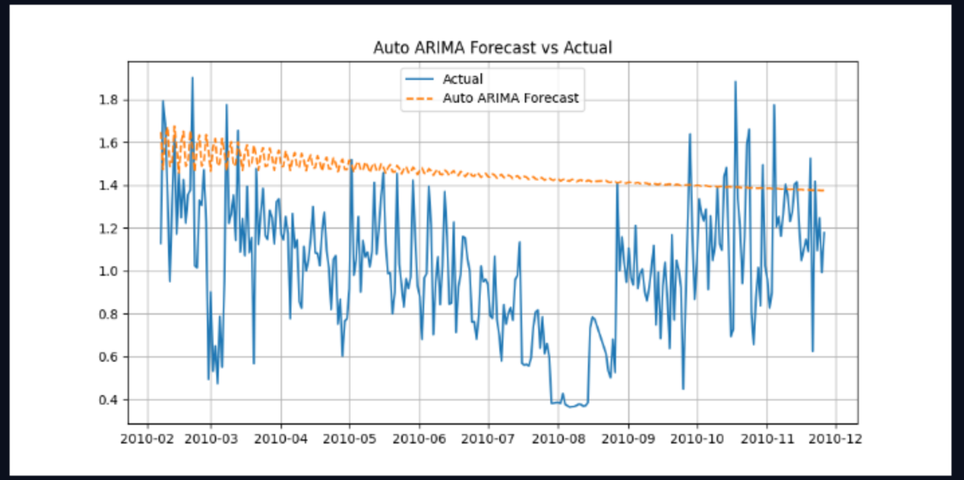
모델 성능 비교 및 분석



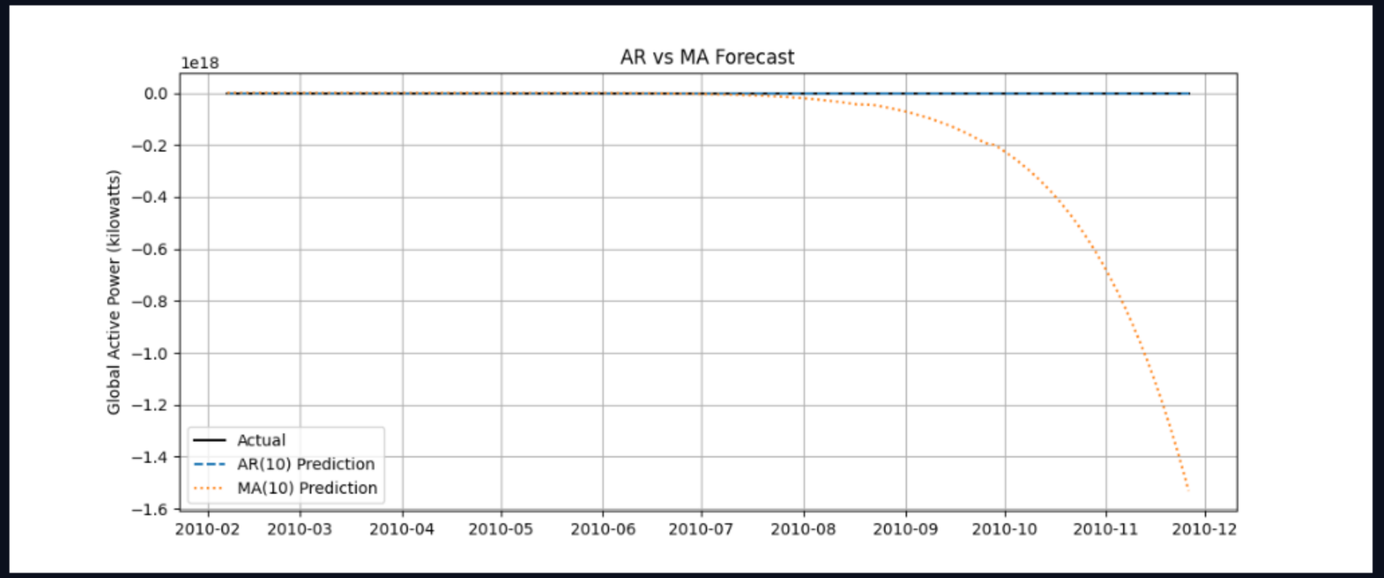
ARIMA 및 Auto ARIMA:

ARIMA는 시계열 데이터의 선형 패턴을 모델링하는 데 적합합니다.
그러나 비선형 패턴을 학습하는 데 한계가 있습니다.
Auto ARIMA는 최적의 파라미터를 자동으로 탐색했지만, 데이터의
비선형성을 충분히 반영하지 못했습니다.





모델 성능 비교 및 분석



AR(10) 및 MA(10):

AR 모델은 과거 값의 선형 조합을 기반으로 예측하며, MA 모델은 과거 오차의 선형 조합을 기반으로 예측합니다.
두 모델 모두 단순한 선형 모델로, 복잡한 시계열 패턴을 학습하는 데 한계가 있었습니다.