결과보고서

과목명	뉴럴 네트워크		
개인/팀여부	□ 개인 ■ 팀(총 2 명)	개인/팀명	TSD
팀구성	박성준(팀장), 백수민(팀원)		
Github URL	https://github.com/qor6/NeuralNetwork_donga/tree/main/Koopa		

결과보고서 작성 안내 사항	
목차	**Ⅰ. 개요** 1. 배경 2. 목적 및 필요성 3. 수행 범위 **Ⅱ. 수행 내용** 1. 수행 절차 2. 수행 내용 및 결과 **Ⅲ. 주요 결과 및 시사점** 1. 주요 결과 요약 2. 결과 활용 및 시사점
작성방향	- 결과보고서는 **30장 내외**로 목차를 준수하여 작성하여야 하며, 필요시 목차 구성에 항목을 추가하여 자유롭게 작성 - 그림 및 도표 등 활용 가능 - 출처 명시(참고 문헌/논문, 이미지, 저자, 사이트 URL 등)
글꼴 및 글자크기	- 본문 글꼴 : 맑은 고딕 - 대분류[1, 2, 3] 항목 : 13포인트(진하게) - 중분류[가, 나, 다] 항목 : 12포인트(진하게) - 소분류[1), 2), 3)] 항목 : 12포인트, 본문내용 : 10포인트

I 개요

1. 배경

시계열 예측은 다양한 응용 분야에서 중요한 주제로 자리 잡고 있다. 금융, 기상, 에너지, 주식 시장 등에서 시계열 예측을 활용한 예측은 정보에 입각한 의사 결정에 핵심적이다. 본 결과 보고서에서의 KooPA 모델은 시계열 데이터 내의 복잡한 패턴을 학습하고 예측하기 위한 혁신적인 접근 방식이라 할 수 있다.

가. 시계열 데이터란?

1) TSD 데이터의 일반적인 특성

TSD(Time Series Data) 분석에서 다루는 Time Series Data는 시간의 순차적인 흐름에 따라 수집되는 데이터 유형을 의미하며, 타임스탬프를 필수적인 요소로 가지는 데이터이다. 공공 분야에서는 날씨와 전력사용량 등을 다루고 민간 분야에서는 주식 가격, 웹 사이트의 일일 방문자 수, 월별 판매량 등에 TSD를 활용한다. 이러한 Time Series Data는 일반적으로 아래와 같은 특성을 가진다.

2) TSD 데이터의 특성

1. 시간의 순서에 따른 종속성: 시계열 데이터는 시간의 순서에 따라서 데이터 간의 종속성이 존재한다. 즉, 이전 시점의 데이터가 다음 시점의 데이터에 영향을 미칠 수 있다. 예를 들어, 날씨는 이전 일자의 날씨에 영향을 받을 수 있다.
2. 계절성: 다수의 Time Series Data가 시간, 연도, 월, 일, 계절 등의 주기에 따른 변동성, 즉 계절성을 가진다. 예를 들어, 여름에는 에어컨 판매량이 증가하고, 겨울에는 에어컨 판매량이 감소하는 등의 에어컨 판매가 계절에 따라 달라지는 패턴을 보일 수 있다.
3. 추세: 시계열 데이터는 시간이 지남에 따라 점차 증가하거나 감소하는 추세를 가질 수 있다. 예를 들어, 경제 성장률, 인구 증가율 등은 시간에 따른 추세 변화가 존재한다.

3) 데이터 분석에서의 TSD 구성 요소

시계열 데이터를 분석할 때는 이러한 특성들을 고려하여야 한다. TSD 분석은 이러한 특성을 분해하여 각 요소를 따로 분석하는 방법을 사용한다. TSD 분석에서 시계열 데이터는 아래 세 가지 구성 요소로 분해한다.

- 트렌드(Trend): 데이터가 장기적으로 증가하거나 감소하는 경향을 뜻한다. 트렌드는 시계열 데이터의 전반적인 방향성을 나타낸다.
- 계절성(Seasonality): 시간의 특정 주기 즉 특정 간격에 따라 반복되는 패턴을 뜻한다. 예를 들어, 연간, 분기, 월간, 주간 등의 주기적 변동을 계절성의 특징을 가진다고 할 수 있다.
- 잔차(Residual): 트렌드와 계절성을 제거한 후 남은 불규칙한 변동을 뜻한다. 이는 예측 가능한 패턴이나 추세를 제외한 무작위 변동을 포함하며, 이상치 탐지 등에 활용할 수 있다.

4) TSD 데이터 분석에 대한 필요성

시계열 예측은 날씨 예보, 에너지 소비, 금융 분석 등 실제 상황에서 매우 중요한 역할을 한다. 이런 예측은 미래의 데이터를 예상하고 계획을 세우는데 필수적이며, 딥러닝 기술의 다양한 데이터

를 사용하는 예측 모델을 만들어 내는 데 큰 성과를 보이고 있다. 그러나 실제 세계의 시계열 데이터는 항상 일정한 패턴을 보이지 않으며 시간이 지남에 따라 변화하는 특성을 가지고 있다. 그래서 시계열 데이터 분석을 통해 미래의 데이터를 좀 더 정확하게 예측하기 위한 방법을 연구를 진행할 필요가 있다.

나. Fore-casting 예측 모델이란?

1) 선행 연구(SOTA)

예측 모델이란 과거 데이터들의 패턴을 기반으로 미래의 값을 예측하는 일을 하는 모델이다. 예측 모델은 수학적 및 통계적 기술을 활용해 시계열 데이터 내의 추세, 패턴, 종속성을 식별하여 시퀀스의 미래 지점을 예측 가능하다. 본 프로젝트에서 선정한 선행 연구 SOTA 논문은 'Koopa: Learning Non-stationary Time Series Dynamics with Koopman Predictors'로, 현재의 시계열 데이터 분석 및 예측을 동적으로 표현하여 예측을 진행하는 모델로 동적 표현을 시계열 데이터에 적용 시켜 현재에도 계속 연구되고 있는 최신 연구에 속하는 예측 모델이다.

2) 관련 연구 동향(Related Work)

시계열 데이터 예측 분야는 고전적으로 통계적 방법을 시작으로 진행되었다. 통계적 방법의 대표적인 예로는 단순한 선형 보간법인 LI(Linear Interpolation)d과 자기 회귀 모델과 이동 평균 모델을 합한 ARIMA(Auto Regressive Integrated Moving Average)가 있었다. 통계적인 방법만으로는 예측의 정확도가 떨어져 시계열 데이터의 비선형적이고 불규칙적인 특성을 학습하고 예측하는 머신러닝 기반 모델이 등장한다. 머신러닝 모델에는 KNN(K-nearest neighbour), SVR(Support vector regression), MLP(Multi Layer Perceptron)이 있다. 또한 머신러닝과 함께 신경망을 활용한 딥러닝 기반 예측 모델도 활발하게 연구되었다. 딥러닝에는 RNN(Recurrent Neural Network), LSTM(Long short term memory models), GRU(Gated recurrent unit)이 있다. 현재는 딥러닝 기반을 넘어서 GPT에 활용된 Transformer를 사용한 예측 모델도 연구되고 있다. 최근에도 존재한 모델들을 경량화 하거나 정확도를 높이는 등의 미래를 예측하기 위한 예측 모델들의 성능을 개선하고 이에 따라 새로운 모델들이 등장하고 있다.

2. 목적 및 필요성

시계열 데이터를 활용한 Koopa 모델 연구를 진행한 목적과 필요성에 대해 설명한다.

가. 전력 데이터 적용의 목적

1) 전력 시장에서의 수요 분석

국제 에너지 기구에 따르면 2015년부터 2018년까지의 전력 수요를 조사해본 결과 2015년에는 전 세계가 278TWh의 수요를 2018년에는 879TWh의 수요로 33%의 전력 수요가 증가하였다. 개발도상국의 강력한 경제 성장 및 지구 온난화로 인해 냉난방용 전력 사용이 증가한 것을 그 원인으로 파악하였다. 이처럼 증가하고 있는 전력 수요를 파악하여 대책을 마련하기 위해서는 EMS(Energy Management Systems)를 통한 전력 분석이 필요하다. EMS는 센서를 통해 전력 데이터를 받고 이를 관리하는 시스템이다. EMS를 활용해 전력 시장의 중요한 측면인 전력 수요를 분석 및 예측하여 증가하는 전력 수요에 따른 문제점들을 파악하고 해결하고자 시계열 데이터에 Koopa 모델을 적용

하는 것이다.

2) 전력 데이터의 문제점

일반적으로 전력 데이터는 시계열 데이터와 같이 비정상성, 계절 변동성, 동적 부하 변경 등의 특징이 있다.

- 비정상성: 전력 소비 패턴은 비정상적 동작을 나타낼 수 있어 기존 예측 모델이 어려울 수 있다.
- 계절 변동: 전력 수요는 날씨, 경제 활동 등의 요인에 영향을 받는 계절적 패턴을 따르는 경우가 많다.
- 동적 부하 변경: 부하 변경의 동적 특성으로 인해 변동 및 갑작스러운 변화에 적응할 수 있는 모델이 필요하다.

이러한 문제점을 가지고 있어 전력 데이터에 활용되는 예측 모델의 정확도 개선 및 경량화를 진행할 필요가 있다.

3) 기존 예측 모델들의 한계점

다양한 데이터를 활용한 딥러닝 기술은 예측 모델을 만들어내는데 큰 성과를 보이고 있다. 그 중에서도 RNN, Attention, Transformer와 같은 딥러닝 기법들이 시간에 따라 변하는 데이터의 패턴을 분석하고 예측하는데 사용되곤 한다. 하지만 실제 세계의 데이터는 항상 일정한 패턴을 보이지 않고 시간이 지남에 따라 특성이 변하는 경향이 있다. 이러한 변동성을 고려한 예측 모델을 만드는 것은 어렵다. 기존의 모델들은 이러한 변동성을 처리 및 고려하지 않고 있다.

나. 전력 데이터 적용의 필요성

1) 결과에 대한 실질적 이점

논문을 통해 확인해 본 결과, 기존 모델들과 비교하여 training time을 77.3% 감소시켜 시계열 데이터 시스템의 학습 효율성을 향상 시켰다. 이로써 기존에 비해 빠른 학습이 가능하며, 실시간 예측 환경에서의 유용성이 부각된다. 또한 메모리 사용량을 76.0%로 절약함으로써 자원의 효율성을 극대화하게 되었다. 이는 메모리 제약이 있는 환경에서도 모델의 성능이 효과적으로 발휘할 수 있음을 의미한다. 이러한 학습 시간 감소와 메모리 절약의 성과는 실제 전력 시장 산업에서의 적용 시 비용 절감과 성능 향상을 의미하며, 특히 전력 데이터에 적용 시에는 에너지 효율성을 증가시키고 비용을 절감하는데 기여할 것으로 기대하고 있다.

2) 결과에 대한 기대 효과

논문에 따르면 이러한 결과는 다양한 응용 분야에서 혁신적인 변화와 혜택을 가져올 수 있습니다. 정확하고 효율적인 예측 모델은 전력 생산과 소비를 최적화 할 수 있다. 이는 에너지 비용의 절감 및 생산 효율의 향상으로 이어질 것이다. 실시간 예측이 가능한 모델은 전력 그리드의 안정성을 높여주고 에너지 관리자들에게 실시간 의사 결정을 지원해준다. 더불어 환경 변화에 민감한 전력 분야에서 미래 예측의 정확성을 향상시키는 것은 전략적으로 중요한 요소이며 새로운 표준을 제시할 것으로 기대된다.

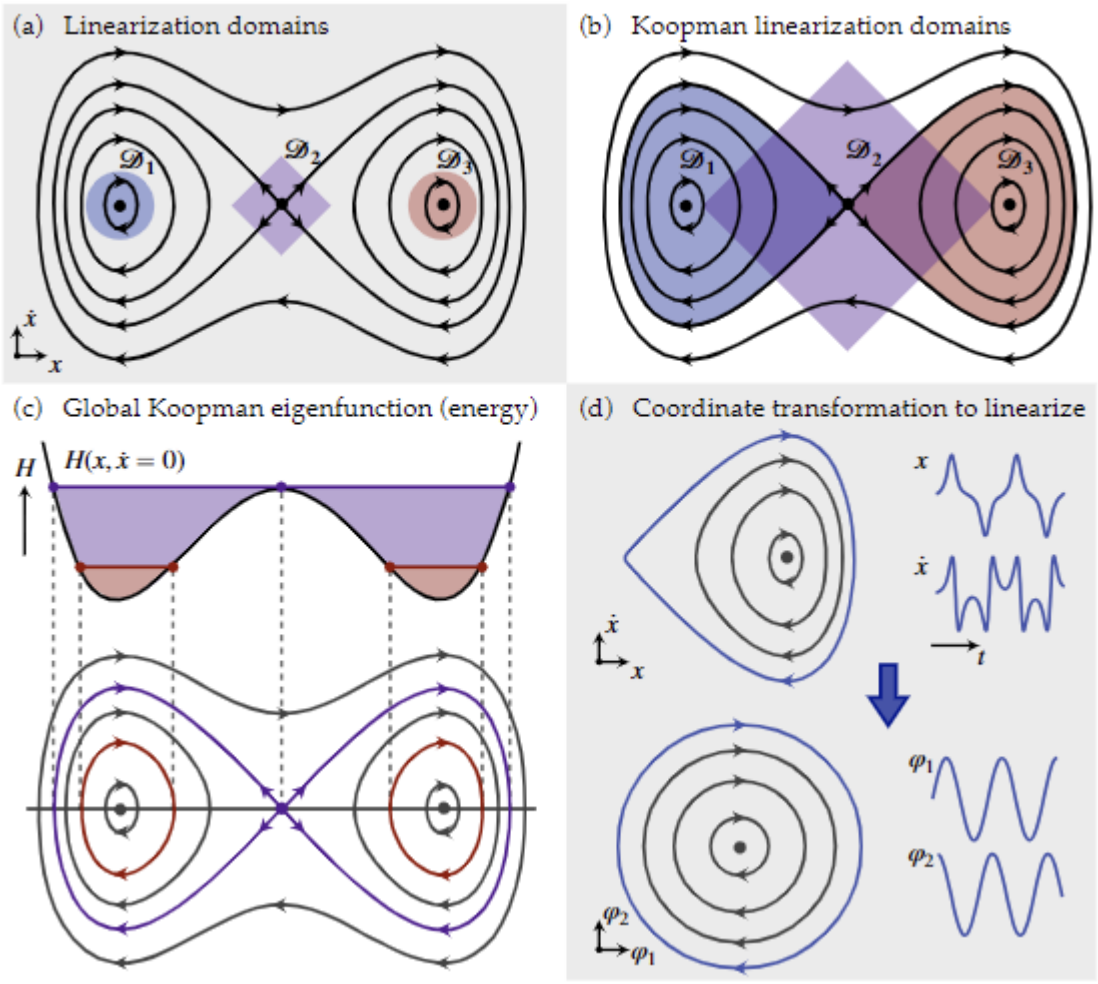
3. 배경 지식

1) Koopman 이론

Koopman이론은 동역학 시스템을 선형 변환으로 모델링하는 수학적인 이론이다. 동역학 시스템의 복잡한 비선형 동작을 선형 변환으로 근사화 할 수 있는 방법을 제공하며 주로 고전 역학이나 통계 역학에서 발생하는 복잡한 동역학 시스템을 다루는데 사용된다.

Koopman 이론을 화학적인 분야로 생각해보면 궤도 함수의 에너지와 이온화 퍼텐셜 에너지의 관계를 나타내는 식이다. 어떤 분자나 원자가 양이온으로 되는 이온화 과정에서 이온화되는 전자로부터 다른 전자들이 영향을 받지 않을 경우 궤도 함수의 에너지와 이온화 에너지는 음수의 관계를 가지게 된다.

Koopman 분석은 3가지가 주요 요소가 된다. 1. 동적 시스템에 대한 고전적이며 기하학적인 접근 방식과 연결하는 엄격한 이론이 존재한다. 2. 접근 방식이 측정 측면에서 공식화되어 빅데이터 및 기계를 활용하는데 이상적이다. 3. 동적 모드 분해와 같은 간단하면서도 강력한 수치 알고리즘 개발 및 확장되어 Koopman 이론을 실제 응용 프로그램에서 실습할 수 있도록 축소되었다.



(a) 고정점 근처의 전통적인 선형화는 시스템이 거의 선형인 작은 영역을 제공한다.
(b) Koopman 이론은 Hartman-Grobman 정리를 확장하여 다음 고정점까지 선형 영역을 확대 가

능하다.

(c) Hamilton에너지와 같은 전역 Koopman고유 함수도 있지만 정답의 위치 정보를 잃는다.

(d) 역학이 hypertoroid에 존재할 때까지 공간과 시간을 재조정하기 위한 좌표 변환을 추구한다.

Koopman이론은 동역학 시스템의 특성을 선형 변환으로 표현하기 위해 Koopman operator를 사용한다. KooPA에서 사용되는 Koopman 이론은 복잡한 비선형 동작을 간단한 선형 동작을 변환하여 동역학 시스템의 복잡성을 이해하고 예측하는 방법을 제공하여 다양하게 응용 가능하다. 최근 딥러닝 확장으로 선형 분석이 가능한 비선형 역학 시스템을 간결하고 해석 가능한 표현을 가능하게 해주었다.

동적 시스템에서의 이산 시간은

$$x_{t+1} = F(x_t)$$

로 표현될 수 있다.

여기서, x_t는 시스템 상태를 나타내고, F는 동적을 설명하는 벡터 필드이다.

이는 비선형성이나 노이즈 데이터 때문에 상태에 직접 시스템 전환이 일어나는 것을 파악하기 어렵다. 그러나, Koopman 이론은 상태가 측정 함수 g의 공간으로 투영될 수 있으며, 무한 차원 선형 연산자 K에 의해 시간적으로 전진하고 조정될 수 있다고 가정한다. 이를 수식으로 표현하면 다음과 같이 표현할 수 있다.

$$K \circ g(x_t) = g(F(x_t)) = g(x_{t+1})$$

Koopman 이론은 유한 차원 비선형 동역학과 무한 차원 선형 동역학 사이의 다리 역할을 한다.
여기서 spectral analysis tool을 적용하여 깊이 있는 분석결과를 얻을 수 있다.

2) 전력 도메인

전력은 전기의 힘을 의미하고 P = VI로, 전압 X 전류로 나타내며 단위는 와트이다.
전압은 전기를 미는 압력으로 단위는 볼트이다. 전류는 전기의 흐름으로 단위는 암페어이다.
이러한 전력 수치를 단위시간으로 곱하게 되면, 전력량의 데이터가 된다.

전력

$$P = V^*I$$

P = 전력(W) / V = 전압(V) / I = 전류(A)

전력량

$$W = V^*I^*t$$
$$W = P^*t$$

W = 전력(Wh) / V = 전압(V) / I = 전류(A) / t = 단위시간(h)

교류 회로에서 소비 전력을 다룰 때는 피상 전력, 유효 전력, 무효 전력의 3가지의 소비 전력을 알아야 한다. 교류 회로의 전원은 주기적으로 변화하면서 에너지를 전달해주다 보니 부하의 종류에 따라 전압과 전류의 위상이 달라지면서 3가지로 나누어지게 된다.

피상 전력(Apparent Power)은 이론상의 전력으로 전압과 전류를 곱한 값을 의미한다. 연결된 부하를 운용하기 위하여 발전소에서 보내야 하는 총 전력으로, 유효전력과 무효전력의 vector적인 합이라고 표현할 수 있다. 단위는 VA로 표시되며 표현식은 아래와 같다.

피상 전력

$$P_{app} = V \times I = I^2 \times Z = \sqrt{(P_a)^2 + (P_r)^2}$$

유효 전력(Active Power)은 유효한 전력 즉, 일을 한 전력을 말한다. 저항에서 소비되는 전력으로 단위는 W이다. 표현식은 아래와 같이 표현된다. 대부분의 전자기기에서 이야기하는 소비전력은 유효전력을 의미한다.

유효 전력

$$P_a = V \times I \cos\theta = I^2 \times R$$

무효 전력(Reactive Power)은 유효전력과는 반대되는 의미로 일을 하지 못한 전력을 말한다. 연결된 부하에서 소비되지 않고, 발전소와 부하 사이를 왔다갔다 하는 전력으로 축적되었다가 방출되는 전력이다. 단위는 VAr이며 표현식은 다음으로 정의된다.

무효 전력

$$P_r = V \times I \sin\theta = I^2 \times X$$

X=반응저항

전력 계수는 피상전력을 유효 전력으로 나눈 값으로 전력의 계수를 나타낸다. 교류 회로에서 전류와 전압의 위상차의 코사인값으로 나타낸다.

역률은 일을 수행하는 동안 소비되는 전력의 비율을 피상전력으로 나눈 값으로 정의한다. 역률가 1이 되는 경우가 가장 이상적인 경우로 피상전력과 유효전력이 같아지는 경우이다. 즉 전력을 보낸 만큼 유효하게 전력이 일을 한 것을 의미한다. 이는 현실세계에서 불가능하여 계속 에너지 효율을 위해 노력해 나가고 있는 부분이다.

4. 수행 범위

가. 데이터 수집

필요한 데이터를 식별하고 수집하는 작업을 진행한다. 데이터의 종류, 규모, 및 품질 등을 고려하여

전력 데이터를 효과적으로 확보한다.

나. 데이터 분석 및 전처리

수집된 데이터를 분석하고 모델 학습에 적합한 형태로 가공하는 과정이다. 이 단계에서는 데이터의 특성을 이해하고 결측치 확인 및 보간 등을 통해 데이터를 정제한다.

다. 모델 구성 및 Training

Koopa 모델을 구성하고 학습시키는 단계이다. Koopa의 구조, 하이퍼파라미터 등을 설정하고 학습 데이터를 활용하여 모델을 훈련시킨다.

라. 모델 튜닝

학습된 Koopa의 성능을 향상시키기 위해 하이퍼파라미터를 조절하고 최적화하는 작업을 진행한다.. 모델의 정확도를 높이기 위한 노력을 진행한다.

마. 결과 분석 및 타 모델과의 비교 분석

최종적으로 얻은 결과를 분석하고, 다른 모델들과의 비교 · 분석을 수행한다. 모델의 성능, 학습 시간, 메모리 사용량 등을 종합적으로 평가하여 Koopa 모델의 우수성을 증명한다.

5. 모델 분석 환경

가. OS

1) Case A: Ubuntu 22.04 LTS
2) Case B: Mac OS 14.3

나. Requirement

0) torch
1) einops==0.4.0
2) matplotlib==3.7.0
3) numpy==1.23.5
4) pandas==1.5.3
5) patool==1.12
6) reformer-pytorch==1.4.4
7) scikit-learn==1.2.2
8) scipy==1.10.1
9) sktime==0.16.1
10) sympy==1.11.1
11) torch==1.7.1
12) tqdm==4.64.1

Ⅱ 수행 내용

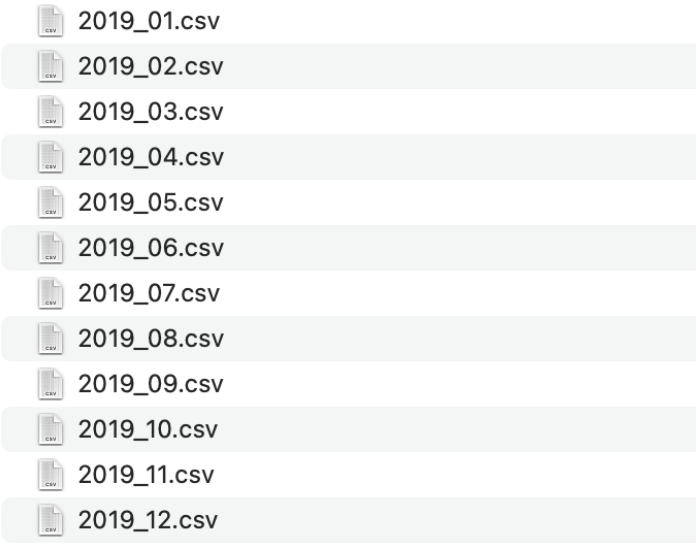
1. 수행 절차

가. 입력 데이터 단계

1) 데이터 수집 및 구성 파악 단계

KooPA 모델을 훈련, 검증, 비교 분석을 수행하기 위해 선정한 데이터는 시계열 데이터 중에서 전력 데이터를 선택하였다. 한국 전력 공사에 공개 데이터 신청을 하여 얻은 경기도의 2019년 연간 전력량 원천 데이터를 활용한다.

본 데이터는 1 ~ 12월 각 월별로 총 12개의 파일로 존재하여, 이를 Concat 하는 작업이 별도로 필요로 했으며, Concat 과정에서 기존 CP949 방식의 Encoding을 UTF-8로 변환하여 1차 결과를 도출하는 등의 Column 구성을 파악하고 파일의 Encoding변환 과정을 진행한다.



1월부터 12월까지의 CSV 데이터 사용

A	B	C	D	E	F	G	H	I	J	K	L
날짜	시간	시/도	읍_면_동	고객번호	계약종별	계약전력	공급방식	고저압구분	유효전력량	지상무효전력량	진상무효전력량
20190301	645	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	1815	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.14	0
20190301	1830	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	730	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	1845	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	745	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	1900	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	800	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.58	0.18	0
20190301	815	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.14	0
20190301	830	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	1915	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.14	0
20190301	1930	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	1945	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	845	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	900	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	2015	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	2000	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.58	0.18	0
20190301	915	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.58	0.18	0
20190301	2045	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0
20190301	2030	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.14	0
20190301	945	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0

3월의 원천 데이터

원천 데이터는 총 12개의 Column으로 구성되어 있으며, 각 Column은 아래의 정보를 포함한다.

Columns			
No.	Column Name (KOR)	Column Name (ENG)	Example
1	날짜	date	20190101
2	시간	time	1000
3	지역	Region	경기도
4	구역	Sub_Region	BB0001동
5	고객번호	Customer_number	b72c577b37
6	계약전력유형	Contract	심야전력(을)III
7	계약전력량	Contract_Power	750
8	공급방식	Supply_type	삼상4선(22.9V-y)
9	고압 / 저압	High-Low	고압
10	유효전력량	Active_Energy	0.58
11	지상무효전력	Lagging_Reactive_Energy	0.18
12	진상무효전력	Leading_Reactive_Energy	0.00

각 Column들의 정보

위 Column을 기반으로 한 달의 row 데이터가 총 680,497개 존재한다.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 680497 entries, 0 to 680496
Data columns (total 13 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   date                    680497 non-null  int64
 1   time                    680497 non-null  int64
 2   Region                  680497 non-null  object
 3   Sub_Region              680497 non-null  object
 4   Customer_number         680497 non-null  object
 5   Contract                680497 non-null  object
 6   Contract_Power          680497 non-null  int64
 7   Supply_type             680497 non-null  object
 8   High-Low                680497 non-null  object
 9   Active_Energy           680497 non-null  float64
 10  Lagging_Reactive_Energy 680497 non-null  float64
 11  Leading_Reactive_Energy 680497 non-null  float64
```

한 달의 각 Column의 개수 확인

위 DataFrame의 간단한 분석 결과에서 Column에 따라 데이터 타입을 확인할 수 있다. 예로, 데이터의 일부를 출력하면 column과 row를 개수도 같이 확인 가능하다.

	date	time	Region	Sub_Region	Customer_number	Contract	Contract_Power	Supply_type	High-Low	Active_Energy	Lagging_Reactive_Energy	Leading_Reactive_Energy
0	20190101	1000	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.58	0.18	0.00
1	20190101	1015	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0.00
2	20190101	1030	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.14	0.00
3	20190101	815	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.18	0.00
4	20190101	830	경기도	BB001동	b72c577b37	심야전력(을)II	750	삼상4선(22.9kV-y)	고압	0.54	0.14	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...
680492	20190131	2230	경기도	BB004동	aaac041cbe	가로등(을)	5	삼상4선(22.9kV-y)	고압	0.83	0.00	0.11
680493	20190131	2245	경기도	BB004동	aaac041cbe	가로등(을)	5	삼상4선(22.9kV-y)	고압	0.83	0.00	0.11
680494	20190131	2315	경기도	BB004동	aaac041cbe	가로등(을)	5	삼상4선(22.9kV-y)	고압	0.82	0.00	0.11
680495	20190131	2130	경기도	BB004동	aaac041cbe	가로등(을)	5	삼상4선(22.9kV-y)	고압	0.83	0.00	0.11
680496	20190131	2300	경기도	BB004동	aaac041cbe	가로등(을)	5	삼상4선(22.9kV-y)	고압	0.83	0.00	0.11

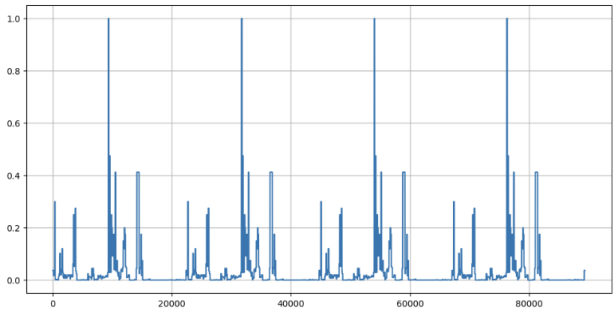
680497 rows × 12 columns

Column과 row 개수 확인

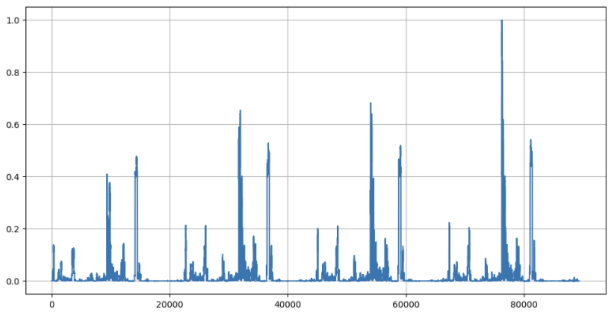
나. 데이터 전처리 단계

1) 데이터 분석

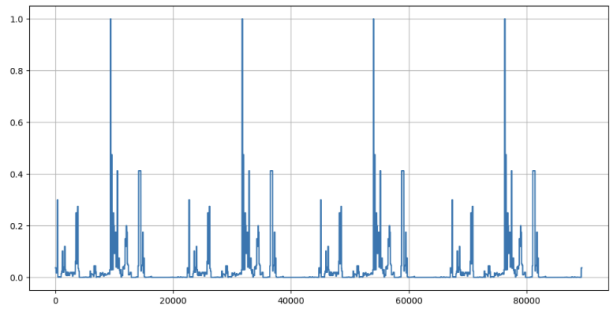
데이터 분석을 진행하기 위해 데이터가 어떤 경향을 보이는지에 대한 형태를 파악해 보아야 한다. 따라서 원천 데이터의 coulmn을 기준으로 나누어 그래프로 그려보면 다음과 같이 그려진다.



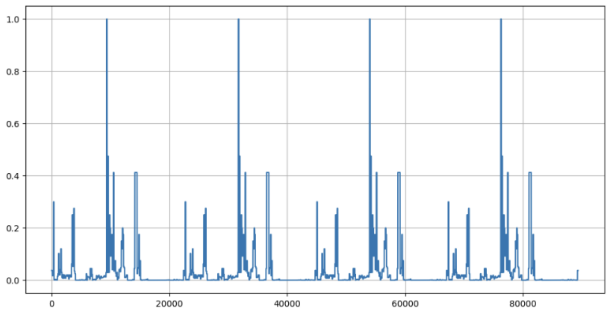
계약전력



유효전력량 (Active_Energy)



지상무효전력량



진상무효전력량

2) 데이터 사용 범위 지정

학기 내 빠르게 진행해야 하는 해당 프로젝트의 특성을 고려하여, 전체 row 중 일부 row를 추출하여 프로젝트를 진행하였다.

date / Active_Energy / Lagging_Reactive_Energy / Leading_Reactive_Energy를 제외한 다른 변수는 동일하게 적용한 값에 해당하는 데이터를 사용하였으며, 데이터의 정보는 아래와 같다.

Data Information		
Period		2019년 01월 01일 00시 00분 ~ 2019년 12월 31일 23시 45분
Freq		15분 간격
공통 사항	Region	경기도
	S_Region	BB001동
	Customer_No	80671b07f8
	Contract	일반용(갑)I고압A
	Contract_P	25
	Supply_t	삼상4선(22.9kV-y)
	H - L	고압
Main Columns		Active_Energy, Lagging_Reactive_Energy, Leading_Reactive_Energy

원천 데이터 중 일부 데이터 범위를 지정

따라서, 표에서 나타내었듯 Active_Energy, Lagging_Reactive_Energy, Leading_Reactive_Energy을 주요 관심 대상으로 두었다.

전체 Column 중 무의미하다고 판단되는 일부 Column을 제외한, 나머지 결과를 편의성의 위해 datetime type 지정, String -> int / float 로 변환하는 작업을 진행하였다.

Contract	일반용(갑)I고압A	⇒	int / 1
Supply_type	삼상4선(22.9kV-y)	⇒	float / 22.9
High - Low	고압	⇒	int / 1

이와 같이 변환을 수행한 수행 결과는 아래와 같으며,

	date	Contract	Contract_Power	Supply_type	High-Low	Active_Energy	Lagging_Reactive_Energy	Leading_Reactive_Energy
0	2019.01.01 0:00	1	25	22.9	1	0.10	0.07	0.00
1	2019.01.01 0:15	1	25	22.9	1	0.08	0.05	0.00
2	2019.01.01 0:30	1	25	22.9	1	0.17	0.17	0.00
3	2019.01.01 0:45	1	25	22.9	1	0.09	0.06	0.00
4	2019.01.01 1:00	1	25	22.9	1	0.08	0.05	0.00
...	...	...	...	...	...	...	...	...
34887	2019.12.31 22:45	1	25	22.9	1	0.10	0.07	0.04
34888	2019.12.31 23:00	1	25	22.9	1	0.12	0.10	0.04
34889	2019.12.31 23:15	1	25	22.9	1	0.08	0.05	0.04
34890	2019.12.31 23:30	1	25	22.9	1	0.13	0.12	0.04
34891	2019.12.31 23:45	1	25	22.9	1	0.10	0.08	0.04

34892 rows × 8 columns

총 34,892 개의 row 값이 존재하는 것을 확인하였다.

3) 데이터 사용 범위 축소 후 추가 데이터 전처리

데이터 사용 범위 축소 후 추가 전처리를 통해 Column을 간소화하고, 코드 처리에 용이하게 변경 하였다.

기존 Column과 비교하여보면 아래와 같은 표를 구성할 수 있다.

Columns			
No.	**기존 Columns**		**전처리 후 Columns**
1	date		**date와 time 데이터를 모두 포함한 1개의 data Column으로 재생성**
2	time		
3	Region		~~(제거)~~
4	Sub_Region		~~(제거)~~
5	Customer_number		~~(제거)~~
6	Contract	⇒	**int형으로 변환하여 데이터 맵핑**
7	Contract_Power		**int형으로 변환하여 데이터 맵핑**
8	Supply_type		**float형으로 변환하여 데이터 맵핑**
9	High-Low		**int형으로 변환하여 데이터 맵핑**
10	Active_Energy		**기존 값 유지**
11	Lagging_Reactive_Energy		**기존 값 유지**
12	Leading_Reactive_Energy		**기존 값 유지**

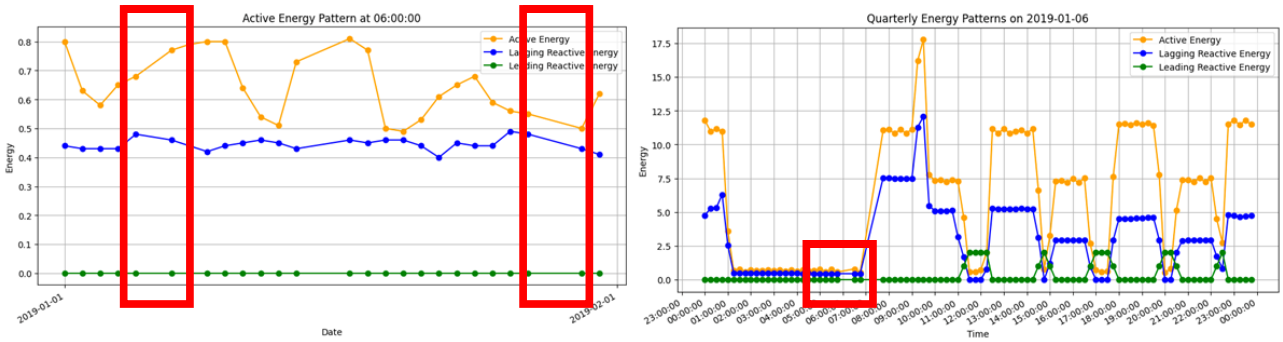
따라서, 최종적으로 데이터 처리를 위한 Columns과 이에 따르는 데이터는 아래와 같다.

No.	Columns Name	Description
1	date	날짜 + 시간 (freq: 15)
2	Contract	계약전력유형
3	Contract_Power	계약전력량
4	Supply_type	공급방식
5	High-Low	고압 / 저압
6	Active_Energy	유효전력량
7	Lagging_Reactive_Energy	지상무효전력
8	Leading_Reactive_Energy	진상무효전력

데이터 분석에 활용할 최종 데이터 Column

앞서 전체 데이터의 row에서 결측치가 없었던 것에 반해, 위 Column에서 최종 도출된 값의 일부를 그래프로 그려보면, 각 컬럼의 일부 값이 결측된 것이 아닌 일부 시간대의 전체 데이터가 누락된 것을 확인할 수 있다.

이는 다음 과정인 결측치 분석 과정에서 자세히 다룬다.



결측값 확인

3) 결측치 분석

전체 row에 대한 결측치 여부를 판단하기 위해 데이터프레임(df)을 15분 단위로 재구성하였다.

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 35040 entries, 2019-01-01 00:00:00 to 2019-12-31 23:45:00
Freq: 15T
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Contract                34892 non-null  float64
 1   Contract_Power          34892 non-null  float64
 2   Supply_type             34892 non-null  float64
 3   High-Low                34892 non-null  float64
 4   Active_Energy           34892 non-null  float64
 5   Lagging_Reactive_Energy  34892 non-null  float64
 6   Leading_Reactive_Energy  34892 non-null  float64
dtypes: float64(7)
```

전처리 후 데이터의 정보 확인

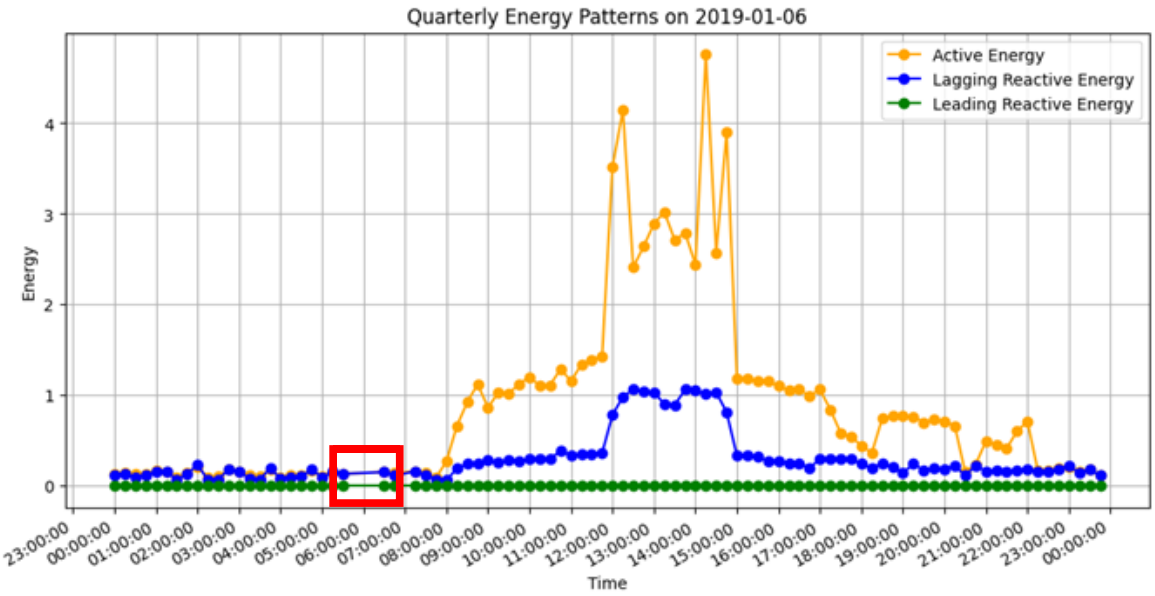
다음은 결측된 값들만 찾아 출력한 것이다. 이를 통해 결측된 값의 개수를 파악할 수 있다.

```
| 2019-06-17 14:30:00 |        nan |        nan |        nan |        nan |        nan |
| 2019-06-17 14:45:00 |        nan |        nan |        nan |        nan |        nan |
| 2019-06-17 15:00:00 |        nan |        nan |        nan |        nan |        nan |
| 2019-06-28 20:30:00 |        nan |        nan |        nan |        nan |        nan |
| 2019-07-16 15:45:00 |        nan |        nan |        nan |        nan |        nan |
| 2019-07-16 16:00:00 |        nan |        nan |        nan |        nan |        nan |
| 2019-07-16 16:15:00 |        nan |        nan |        nan |        nan |        nan |
| 2019-08-15 18:00:00 |        nan |        nan |        nan |        nan |        nan |
| 2019-12-04 15:30:00 |        nan |        nan |        nan |        nan |        nan |
누락된 timestamp의 개수: 148
```

2019년 01월 01일 00시 00분부터 2019년 12월 31일 23시 45분까지의 데이터 기대값은 35040개 이나, 현재 34892개의 데이터만 존재하는 것을 확인할 수 있다. 즉 결측이 된 값이 존재한다는 것을 의미한다.

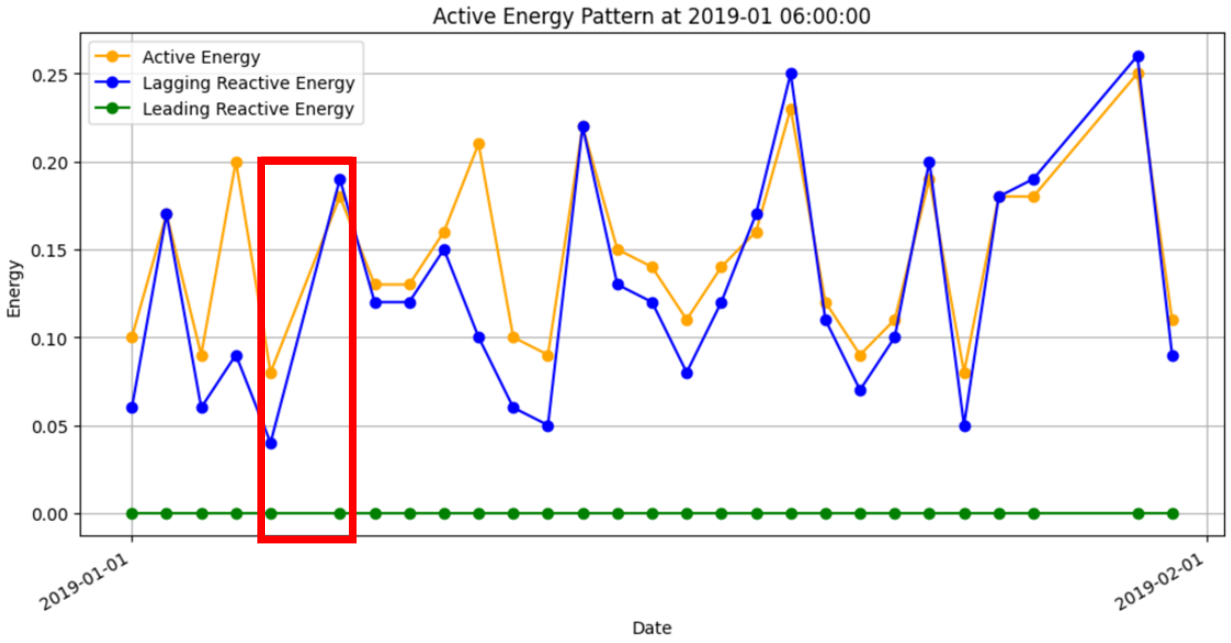
* 기대값: 35,040개 (15분 간격 * 365일 * 24시간)
* 실제값: 34,892개 (데이터 누락 148개, 전체 데이터 중 0.42%)

결측치를 포함한 데이터를 예로 들어, 2019년 1월 6일의 하루 간의 데이터를 그래프로 형상화하였을때, 아래와 같이 결측치를 확인할 수 있다.



하루 결측값 확인

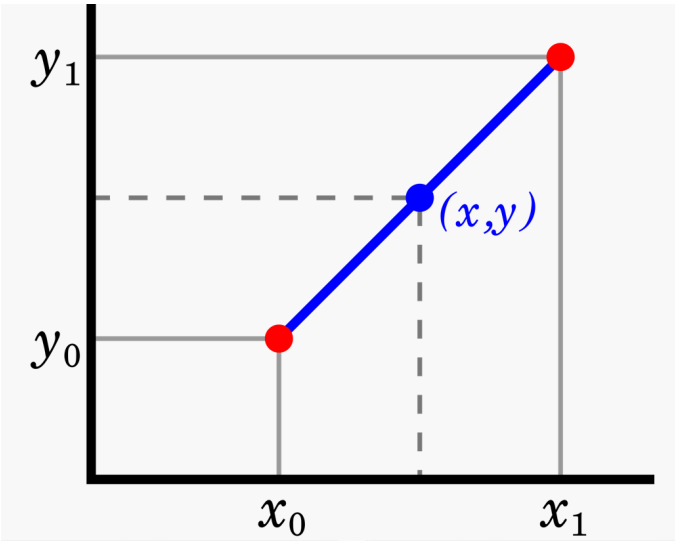
또한 2019년 1월 간 06시 데이터를 그래프로 형상화하였을때, 아래와 같이 결측치를 확인할 수 있다.



한 달 결측값 확인

4) 결측치 보간

앞서 분석한 결측데이터를 보간하기 위해 Linear interpolatation을 수행한다.
Linear interpolation은 주어진 두 점 사이의 직선 상에서 새로운 값을 추정하는 보간 기법 중 하나이다. 이는 두 점을 연결하는 선분 상에서의 위치에 따라 값을 선형적으로 추정하는 방법이다. 주로 두 간격으로 나누어진 두 점에서 다른 위치에 대한 값을 추정할 때 사용한다.



Linear Interpolatation

Linear Interpolation의 수행과정을 수식을 통해 일반화 하면 다음과 같다.

$$
\begin{aligned}
y &= y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0} \\
&= \frac{y_0(x_1 - x_0)}{x_1 - x_0} + \frac{y_1(x - x_0) - y_0(x - x_0)}{x_1 - x_0} \\
&= \frac{y_1 x - y_1 x_0 - y_0 x + y_0 x_0 + y_0 x_1 - y_0 x_0}{x_1 - x_0} \\
&= \frac{y_0(x_1 - x) + y_1(x - x_0)}{x_1 - x_0}
\end{aligned}
$$

위의 formula를 바탕으로 연산하여 결측치가 보간되어 전처리가 진행된 데이터의 도출 결과이다.

2019-12-04 14:15:00	1	25	22.9	1	0.09	0.07	0.03
2019-12-04 14:30:00	1	25	22.9	1	0.04	0.01	0.04
2019-12-04 14:45:00	1	25	22.9	1	0.15	0.15	0.03
2019-12-04 15:00:00	1	25	22.9	1	0.06	0.03	0.04
2019-12-04 15:15:00	1	25	22.9	1	0.07	0.04	0.04
2019-12-04 15:30:00	1	25	22.9	1	0.1	0.08	0.04
2019-12-04 15:45:00	1	25	22.9	1	0.13	0.12	0.04
2019-12-04 16:00:00	1	25	22.9	1	0.03	0	0.05
2019-12-04 16:15:00	1	25	22.9	1	0.15	0.15	0.03
2019-12-04 16:30:00	1	25	22.9	1	0.06	0.03	0.04

결측치가 보간된 데이터

위의 일련의 과정을 통해 결측치를 정상적으로 보간하여 365일 간 15분 단위 데이터를 완성하였다.

다시 한 번 누락된 timestamp 개수를 출력해보면 모두 정상적으로 보간되어 결측값이 0임을 확인 할 수 있다.

```
누락된 timestamp:
 Empty DataFrame
Columns: [Contract, Contract_Power, Supply_type, High-Low, Active_Energy, Lagging_Reactive_Energy, Leading_Reactive_Energy]
Index: []

누락된 timestamp의 개수: 0
```

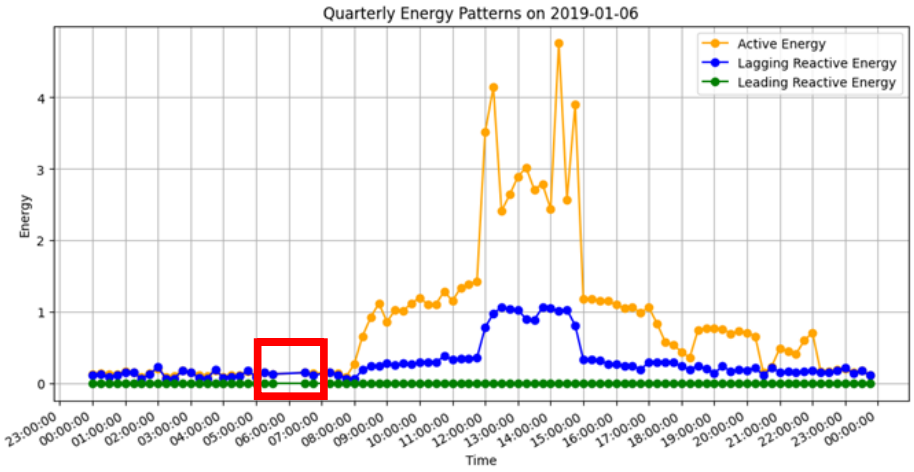
결측된 값이 0로 없다

전처리로 결측치 보간을 진행 한 후 최종 데이터 구성을 살펴보면 아래와 같다.

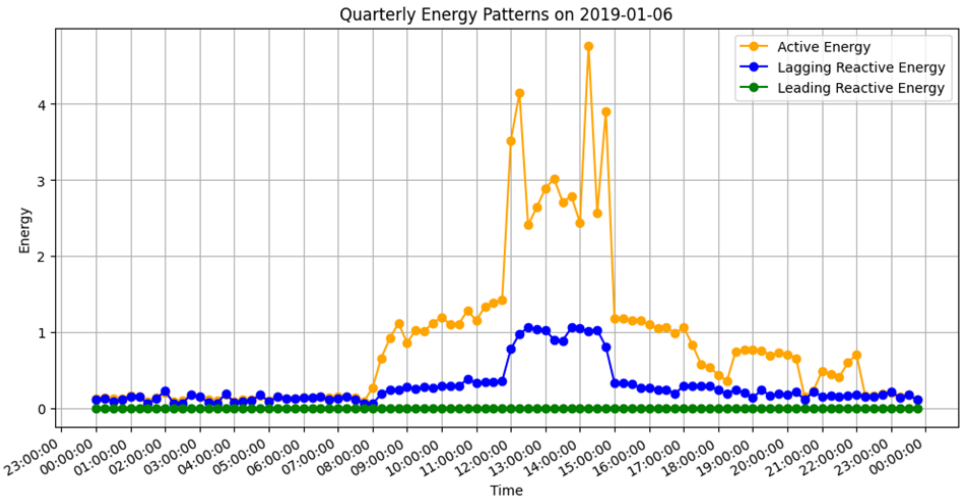
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 35040 entries, 2019-01-01 00:00:00 to 2019-12-31 23:45:00
Freq: 15T
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Contract                35040 non-null  float64
 1   Contract_Power          35040 non-null  float64
 2   Supply_type             35040 non-null  float64
 3   High-Low                35040 non-null  float64
 4   Active_Energy           35040 non-null  float64
 5   Lagging_Reactive_Energy  35040 non-null  float64
 6   Leading_Reactive_Energy  35040 non-null  float64
dtypes: float64(7)
memory usage: 2.1 MB
None
```

최종 데이터 구성

아래의 그래프로 결측치가 보간된 것을 확인하면 값이 채워진 것을 비교하여 확인할 수 있다.



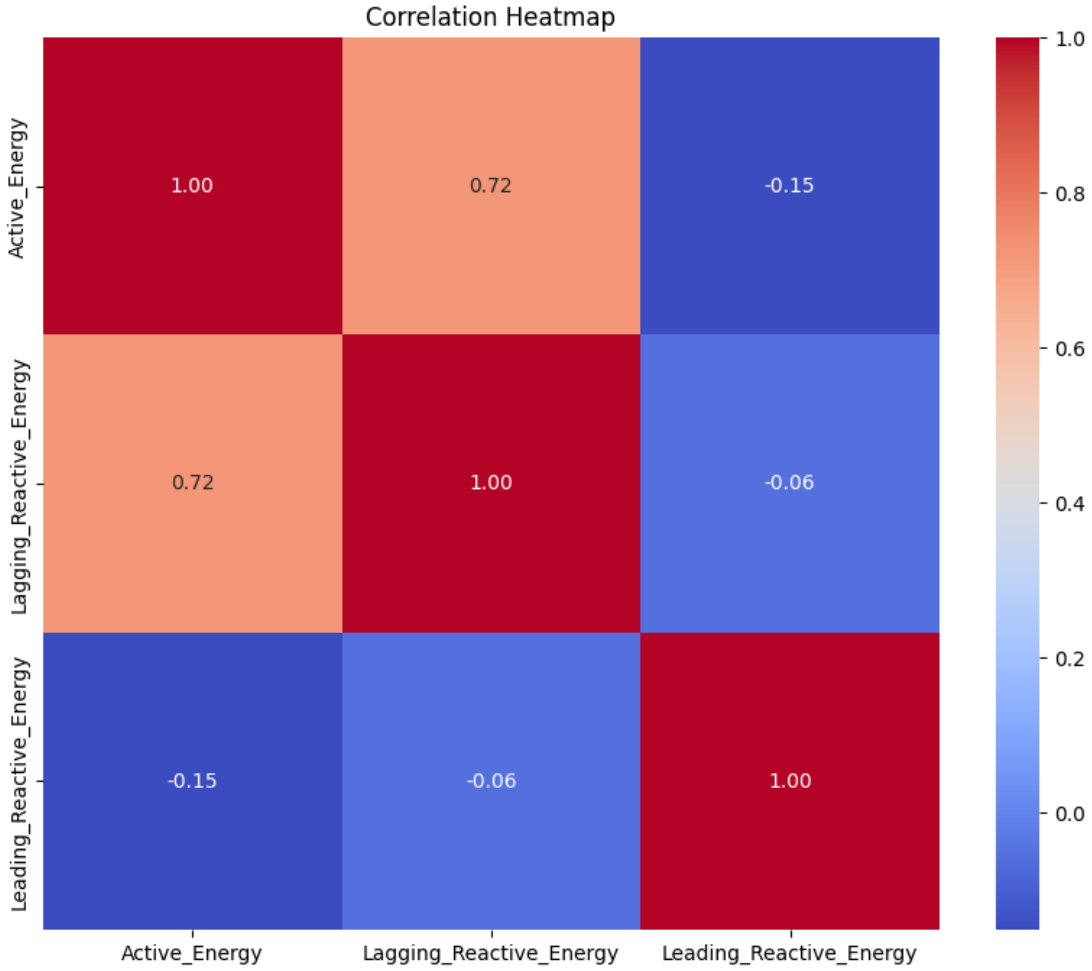
결측치 보간 전



결측치 보간 후

5) 상관관계 분석

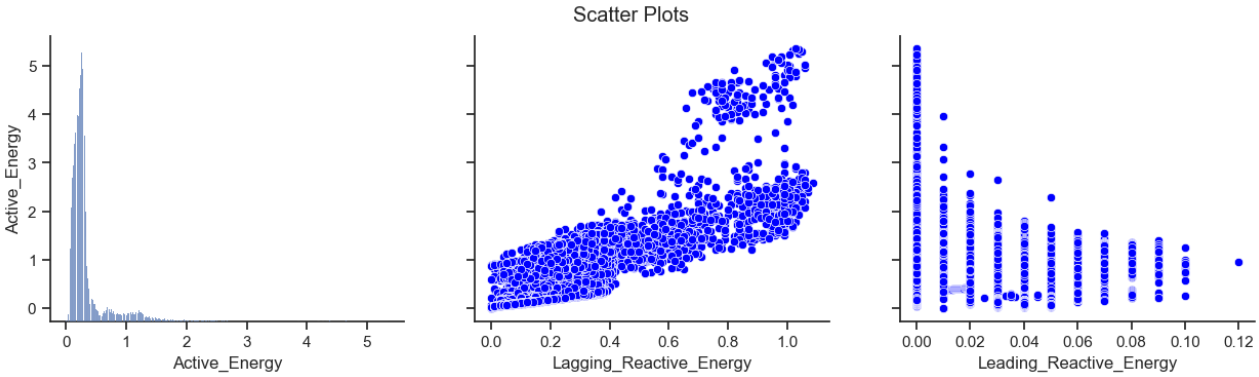
히트맵을 통해 상관관계를 분석한 결과이다.



주요 Column의 히트맵

이를 통해, 비교적 Lagging_Reactive_Energy와 Active_Energy의 상관성이 높은 것을 확인하였고, Leading_Reactive_Energy와 Lagging_Reactive_Energy는 상관성이 적은 것으로 확인할 수 있다.

다음은 Plot을 통해 Active_Energy / Lagging_Reactive_Energy / Leading_Reactive_Energy의 산점도를 나타낸 그래프이다.



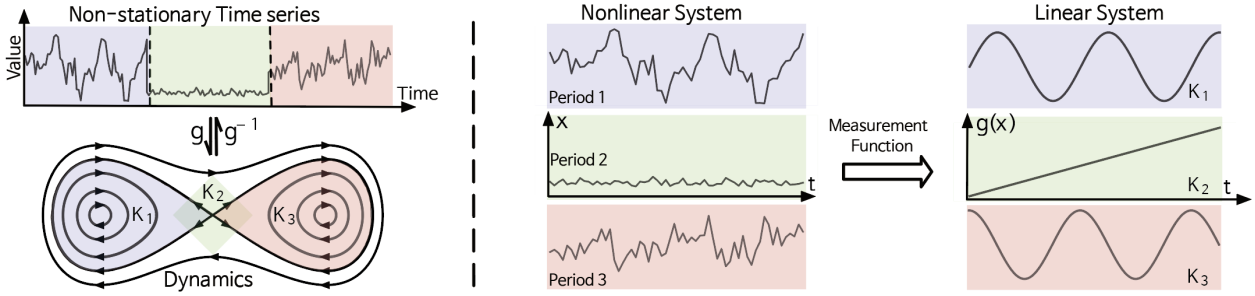
주요 Column의 산점도 Plot 그래프

다. 모델 학습 단계

1) KooPA 설명

KooPA는 Koopman 이론을 활용해 비정상적인 시계열 데이터를 처리하기 위한 방법으로 제공된다. KooPA는 여러 개의 Koopa Block으로 구성되어 있어 입력 데이터의 동적을 학습한다. 이전 블록에서 학습된 동적의 잔차를 입력으로 받아 계층적으로 동적 학습 즉 연산이 수행된다. 여러 Koopa Block들의 학습을 거친 후 여러 Block의 합을 통해 모델이 값을 예측한다.

Koopa Block은 Fourier Filter을 거쳐 동적 예측이 진행되는데 여기서 Fourier Filter는 주파수 도메인 통계를 활용하여 비정상적인 시계열 데이터를 시간에 따라 변하지 않는 부분인 시간 불변동성 성분과 시간에 따라 변하는 시간 변동성 성분으로 분리한다. 주파수 도메인 통계는 시계열 데이터의 주요 특성을 이해하고 그를 기반으로 구별하는데 사용한다. 분리하는 이유는 데이터의 다양한 성격을 고려하여 예측 모델을 더 정확하게 구축할 수 있도록 해주기 때문이다. 이후 분리되어 있는 구성요소들을 예측하기 위해 Koopman Predictor를 설계하고 이 Predictor를 여러 층으로 쌓을 수 있도록 하여 Block을 구성한다. Fourier Filter로 나누어진 시간-불변동성 데이터는 Koopman embedding을 활용해 시계열 데이터를 고차원 공간으로 매핑하여 측정 함수를 구성함으로써 복잡한 동적을 쉽게 분석 가능해진다. 시간-변동성 데이터는 Koopman operator를 사용해 시계열 데이터의 시스템 경향을 설명하는 선형 연산자로 시간에 따라 변하는 부분의 강한 지역성을 다루기 위해 문맥에 맞게 계산하며 실제 관측을 활용하여 예측의 시간 범위를 더 넓게 확장 가능하다. Koopman Predictor에서의 결과를 재구성하여 손실을 해결하고 최종 예측 목표를 MLP를 통해 최적화 한다.



KooPA 이론

Fourier Filter는 각 블록의 시작에서 수행한다. 먼저, 주어진 입력 데이터 X(b)에 Fourier Filter를 적용하여 두 부분으로 나눕니다. X(b)var: 시간에 따라 변하는 변동성 부분 (Time-variant), X(b)inv: 불변동성 부분 (Time-invariant) 로 나누어진 부분에 대해 각각의 Koopman Predictor (KP)가 예측을 수행합니다.

$$X_{var}^{(b)}, X_{\in v}^{(b)} = FourierFilter(X^{(b)})$$

X(b)var KP는 지역적으로 해석된 연산자 솔루션을 계산한다. 이때 시계열 데이터는 일정한 기간 동안의 segment로 나누어져 snapshot으로 배열된다. 이 지역적인 연산자 솔루션은 lookback 창 내에서 시간에 따라 변하는 동역학을 capture하기 위해 사용된다.

X(b)inv KP는 모델 파라미터로 설정된 연산자를 사용하여 lookback-forecast 창에서 전역적으로 학습된다. 전체 시계열 데이터를 고려하여 예측에 필요한 전역적인 동역학을 학습하는 것으로 파악된다.

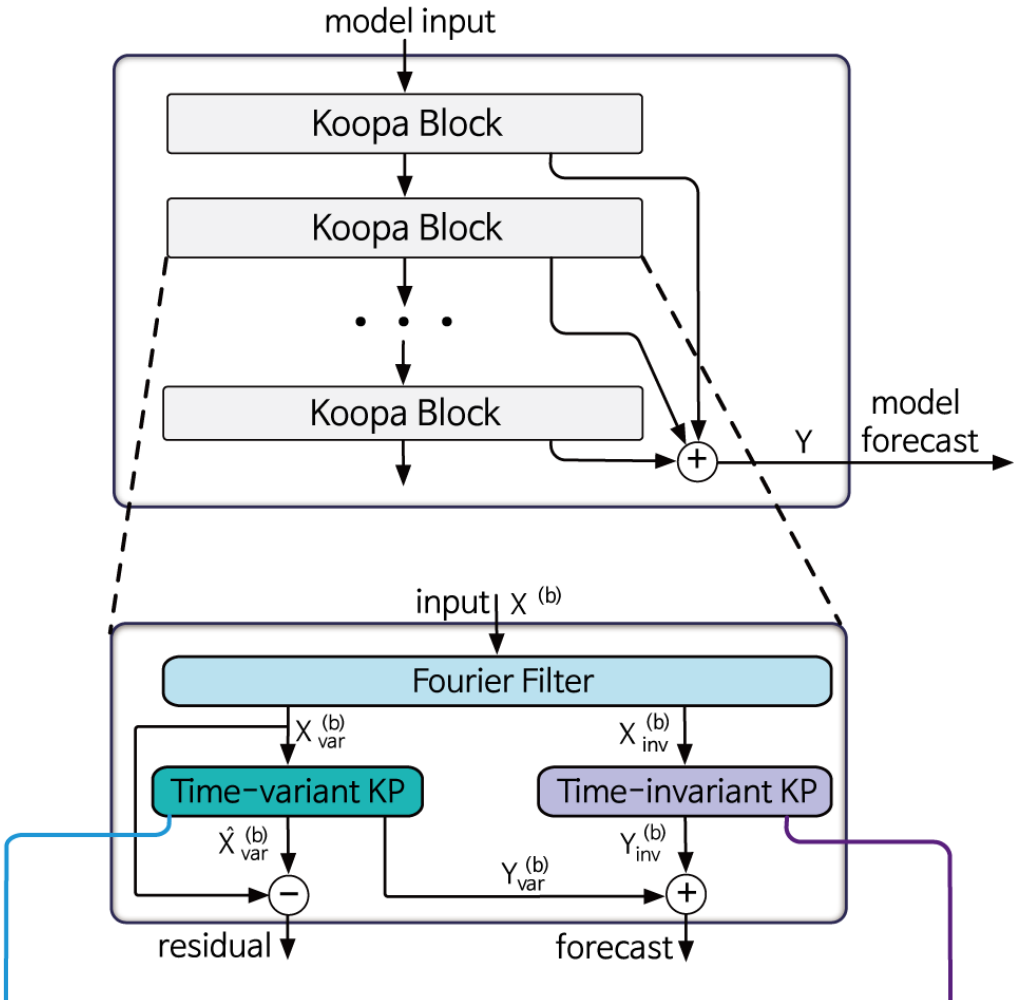
$$\widetilde{X_{var}^{(b)}},\ Y_{var}^{(b)} = TimeVarKP(X_{var}^{(b)})$$

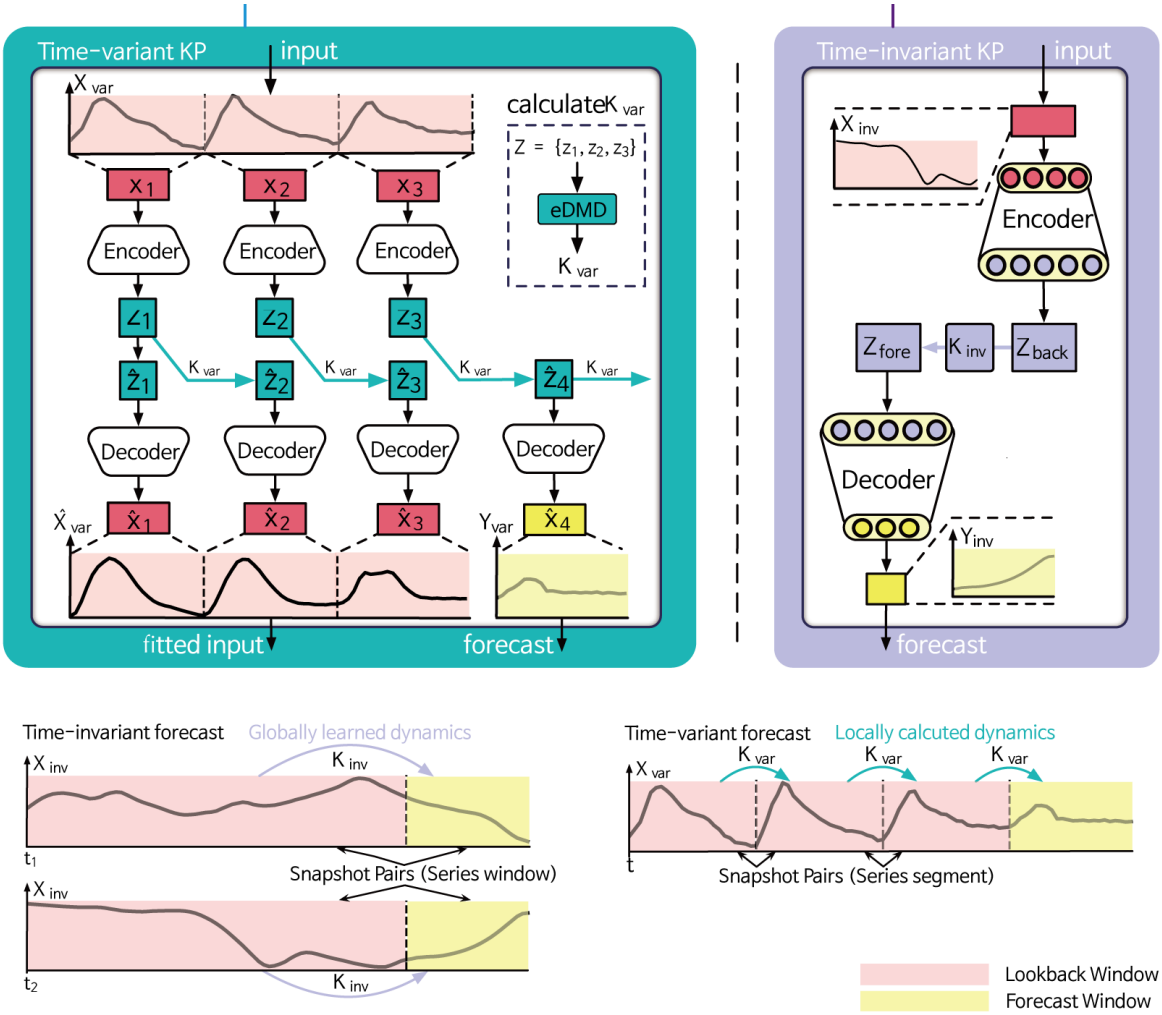
Time-variant KP (시간 변동 KP)는 입력 X(b)var을 사용하여 예측을 수행한다.
이때, Time-variant KP는 동시에 fiting된 입력 X^(b)var을 출력한다.
이것은 시간에 따라 변하는 부분에 대한 예측과 fiting된 입력이다.

$$Y_{\in v}^{(b)} = TimeInvKP(X_{\in v}^{(b)})$$

Time-invariant KP (시간 불변 KP)는 입력 X(b)inv을 사용하여 예측을 수행한다.
이것은 일정한 부분에 대한 예측입니다.

다음은 앞선 설명한 KooPA의 구조도이다.





이러한 방식으로 Fourier Filter와 각각의 Koopman Predictor로 입력 데이터를 적절히 분리하고 예측에 활용합니다. 이렇게 함으로써 Koopa는 시계열 데이터의 다양한 동적 특성을 효과적으로 분석하고 예측할 수 있다. Koopa 모델은 기존의 Koopman Autoencoders (KAEs)와는 다르게 엄격한 재구성을 위한 손실 항을 도입하지 않는다. 대신, 다음 Blok의 학습을 위해 잔차인 X(b+1)를 다음 Block의 입력으로 사용하여 보정 연산자를 학습한다. 그리고 모델이 예측한 결과 Y는 모든 Koopa Block에서 수집한 예측 구성 요소 Y(b)var, Y(b)inv의 합으로 구성된다.

2) KooPA의 효과

향상된 예측 정확도: Koopa는 비정상 시계열 역학을 효과적으로 처리하도록 설계되어 기존 모델에 비해 예측 정확도가 향상되었다는 것이다. Koopman 이론을 활용하고 구성요소를 분리함으로써 Koopa는 복잡하고 진화하는 시계열 데이터에 대해 보다 정확한 예측을 제공할 수 있다.

Train 효율성 향상: Koopa 모델은 training time 측면에서 효율성 향상을 보여준다. Koopman 이론과 제안된 모델 구조를 사용하면 Koopa는 경쟁력 있는 성능을 유지하면서 훈련 시간을 크게 줄일 수 있다. 이러한 효율성은 더 빠른 모델 개발 및 배포에 기여할 수 있다.

메모리 사용량 최적화: Koopa는 메모리 절약을 달성하여 메모리 사용량을 76.0% 줄인다. 이는 특히 리소스가 제한된 환경에서 중요한 효과로, Koopa를 비정상 시계열을 처리하기 위한 리소스 효율적인 솔루션으로 만든다.

시계열 역학의 분리: Koopa는 푸리에 필터와 Koopman 예측기를 사용하여 시계열 데이터에서 구성 요소를 분리하는 새로운 접근 방식을 도입한다. 분리 효과로 기본 역학을 더 명확하게 이해할 수 있어 복잡한 시계열 패턴을 더 잘 해석하고 분석할 수 있다.

모듈식 및 계층적 학습: Koopa는 쌓을 수 있는 Koopa 블록으로 구성되며, 각 블록은 이전 블록의 피팅된 역학의 잔차를 입력으로 사용하여 계층적 역학을 학습한다. 이 모듈식 및 계층적 학습 구조는 시계열 데이터의 복잡한 패턴을 포착하고 표현하는 모델의 능력에 기여한다.

다양한 역학에 대한 적응성: Koopa 모델은 시계열 데이터 내의 다양한 역학에 적응하도록 설계되었 있어 시간적 이웃에서 상황 인식 연산자를 계산함으로써 구성 요소가 나타내는 강력한 지역성을 효과적으로 처리할 수 있다. 따라서 다양하고 진화하는 패턴을 지닌 광범위한 실제 애플리케이션에 적합 가능하다.

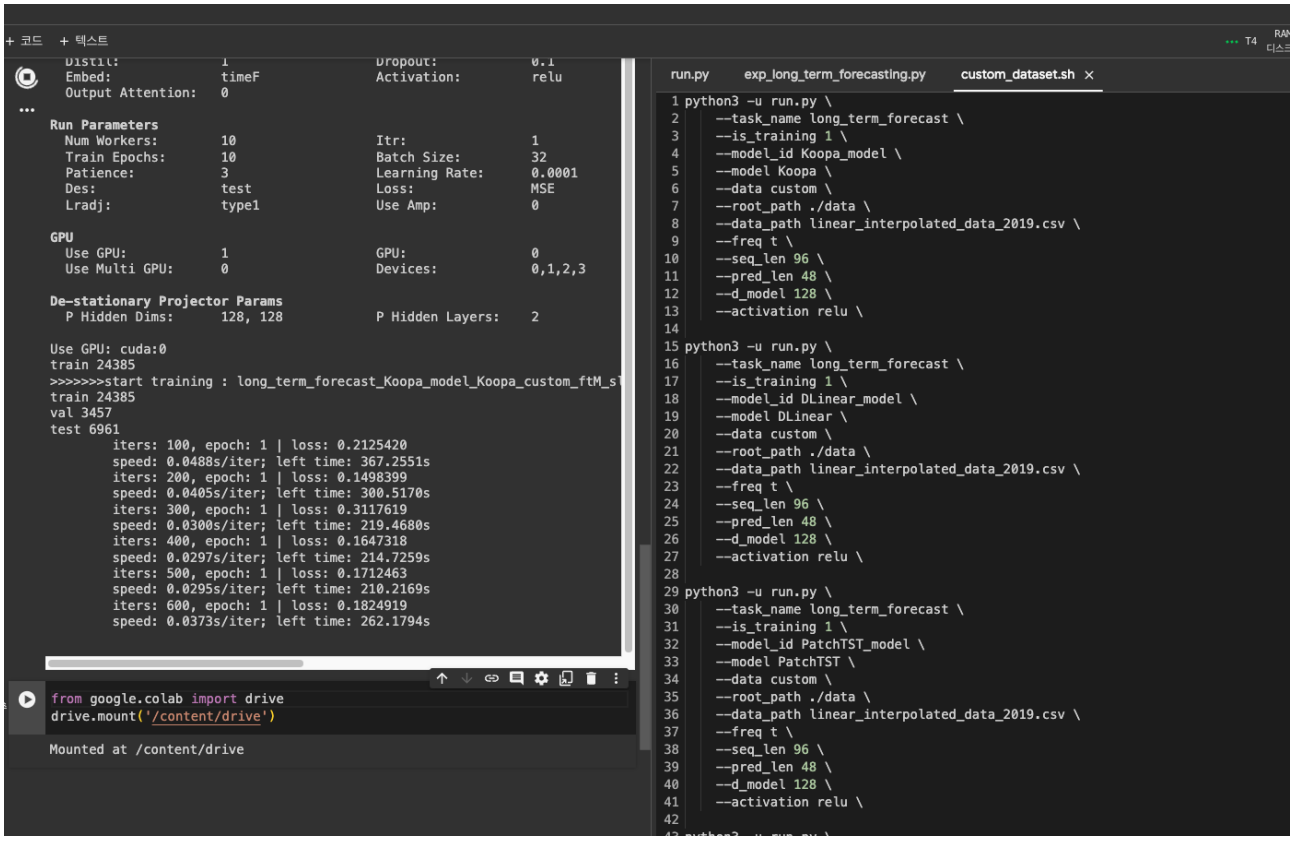
라. 훈련 및 검증 단계

1) 훈련 단계

앞서 소개했던 모델 분석 환경 [Case A: Ubuntu (GPU 가속) / Case B: Mac os (CPU 가속)]을 기반으로 분리하여 각각 동일한 코드로 실험을 수행하였다.

KooPA의 경우, CPU 가속 환경에서도 비교적 빠른 속도로 결론을 도출하였으며, 메모리 사용량 또한 타 모델에 비교하여 낮은 것을 확인할 수 있었다.

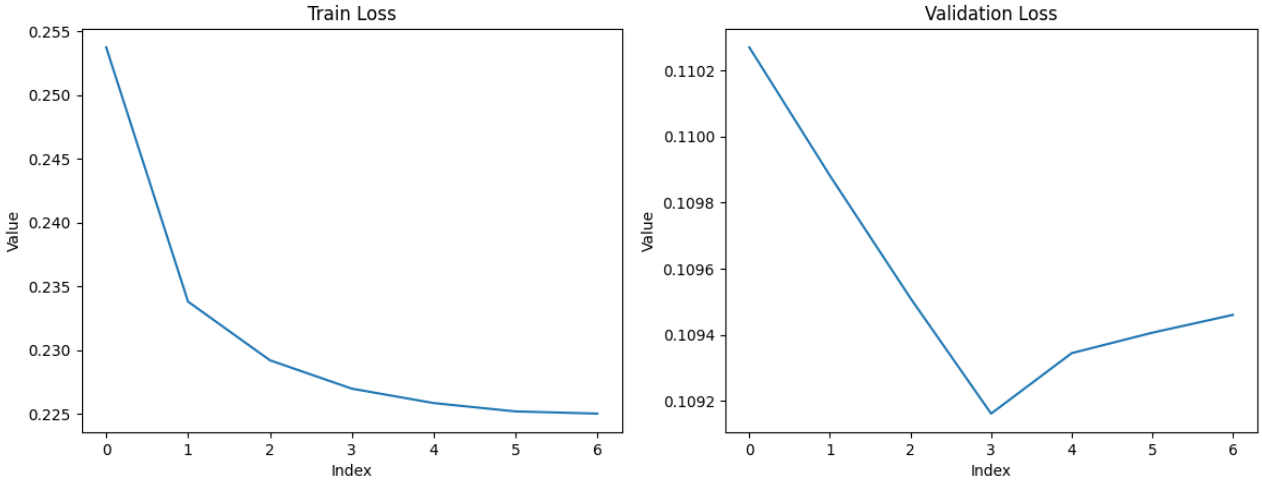
훈련을 여러 번 시도하여 가장 좋은 성능을 나타내는 지표를 저장하였다. 보다 효율적인 hyperparameter tuning과 모델 비교를 위해 각 조건별 요소들을 쉘스크립트(*.sh)파일 하나에 입력하여 순차적으로 학습과 테스트 절차를 수행하였다.



```
GPU: cuda:0
>>>>start training : long_term_forecast_TimesNet_model_TimesNet
n 24385
3457
 6961
    iters: 100, epoch: 1 | loss: 0.2548615
    speed: 10.8530s/iter; left time: 81625.1282s
```

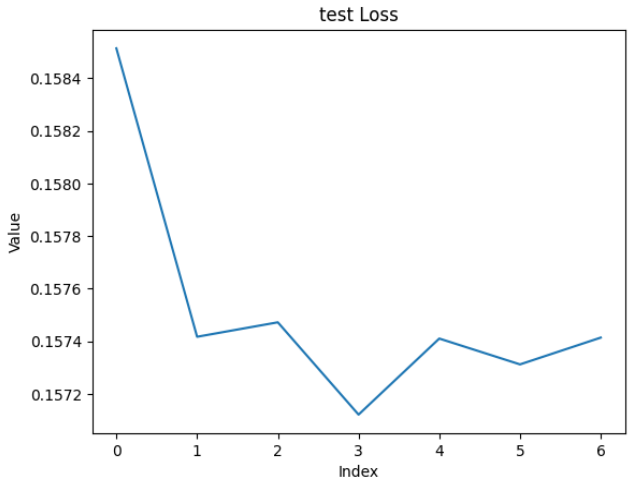
```
Epoch: 7 cost time: 35.27449417114258
Epoch: 7, Steps: 762 | Train Loss: 0.2250287 Vali Loss: 0.1094600 Test Loss: 0.1574138
EarlyStopping counter: 3 out of 3
Early stopping
>>>>>>>testing : long_term_forecast_model_1_Koopa_custom_ftM_sl96_ll48_pl48_dm128_nh8_el2_dl1_df2048_fc1_ebtimeF_dtT
test 6961
test shape: (6961, 1, 48, 7) (6961, 1, 48, 7)
test shape: (6961, 48, 7) (6961, 48, 7)
mse:0.15712106227874756, mae:0.15051478147506714
```

다음은 train과 validation loss 값을 그래프로 그려낸 것이다.



2) 검증 단계

test데이터를 통해 loss 값을 확인한 후 검증을 진행한다.



2. 수행 내용 및 결과

가. 수행 내용

1) 모델 성능 지표

모델의 성능 지표를 측정할 필요가 있다. 대표적인 성능 지표로는 MSE, MAE가 있다.

MSE(Mean Squared Error)는 실제 값에서 예측 값을 뺀 오차에 제곱을 해서 평균을 낸 지표로 오차의 면적의 합을 의미한다. 이 값이 낮으면 낮을수록 좋은 모델이라고 말할 수 있다. 그러나 MSE의 경우 값을 제곱하기 때문에 절댓값이 1미만인 값은 더 작아지고 1보다 큰 값은 더 커지는 왜곡이 발생한다. 따라서 특이값의 영향을 많이 받는다.

MAE(Mean Absolute error)는 실제 값에서 예측 값을 뺀 오차들의 평균 절댓값을 낸 지표로 전체 데이터의 학습된 정도를 쉽게 파악할 수 있다. 그러나 어떤 식으로 오차가 발생했거나 절댓값을 취하기 때문에 음수인지 양수인지 알 수가 없다.

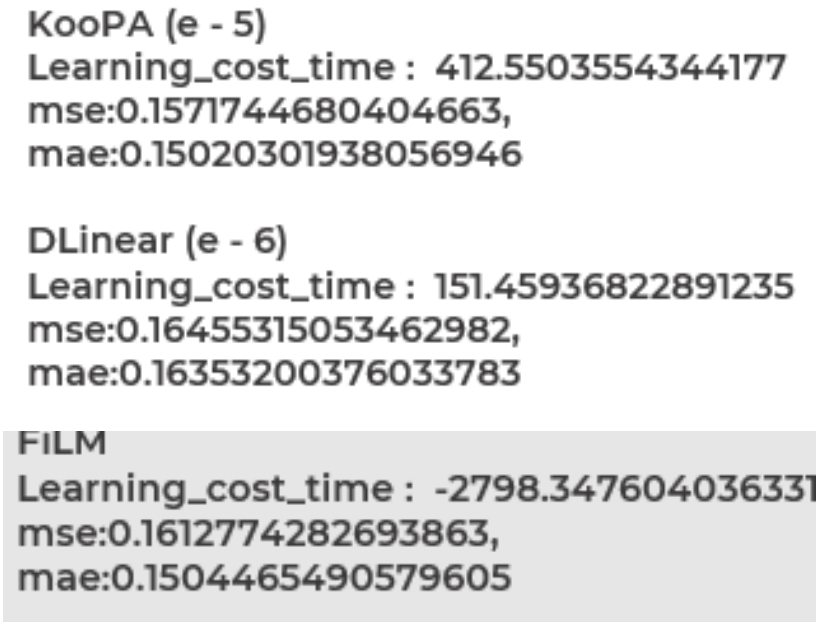
2) 모델 성능 지표에 따른 모델 평가

Koopa 모델을 돌린 결과 MSE는 0.157이며, MAE는 0.15로 나왔으며 Learning time은 412.5초로 나타나 높은 정확도와 낮은 학습시간을 보여준다.

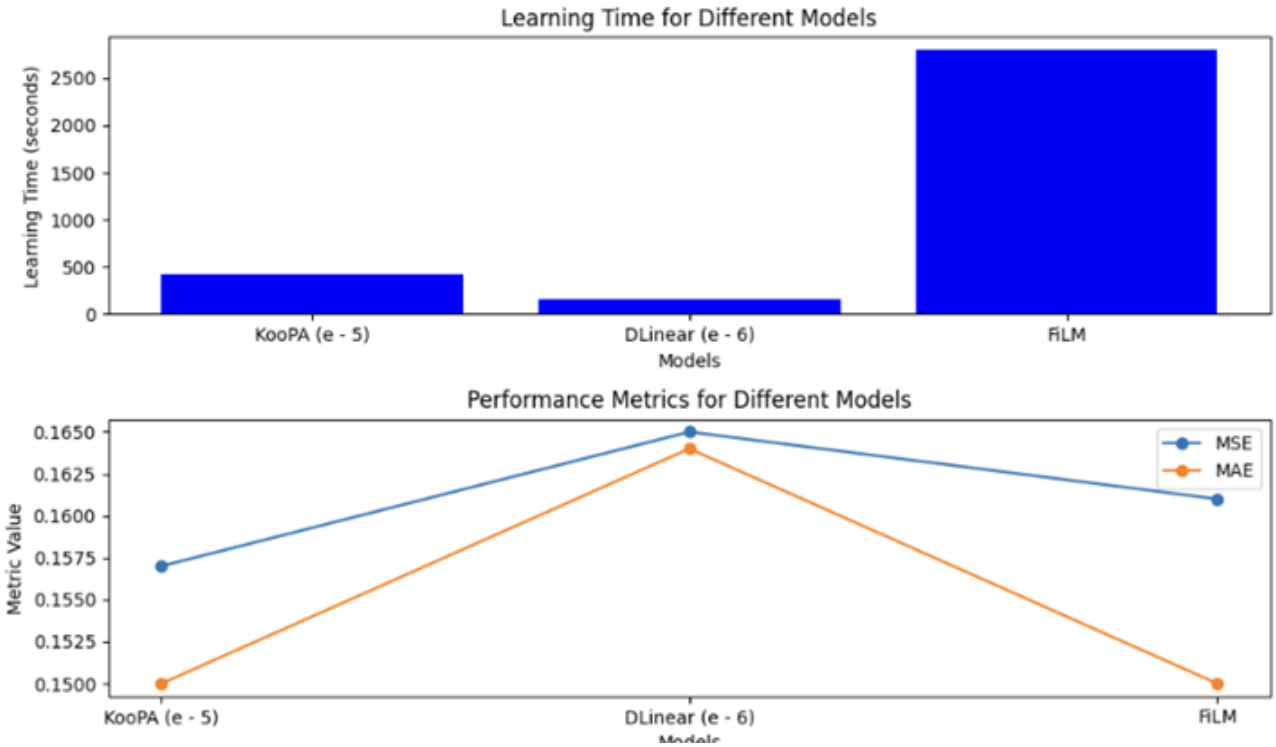
나. 수행 결과

1) 타 모델과의 비교

KooPA와 타 long-term forecasting model과의 비교를 해본 결과 중 비교적 MSE / MAE 값이 낮은 모델과의 비교를 수행하였다.

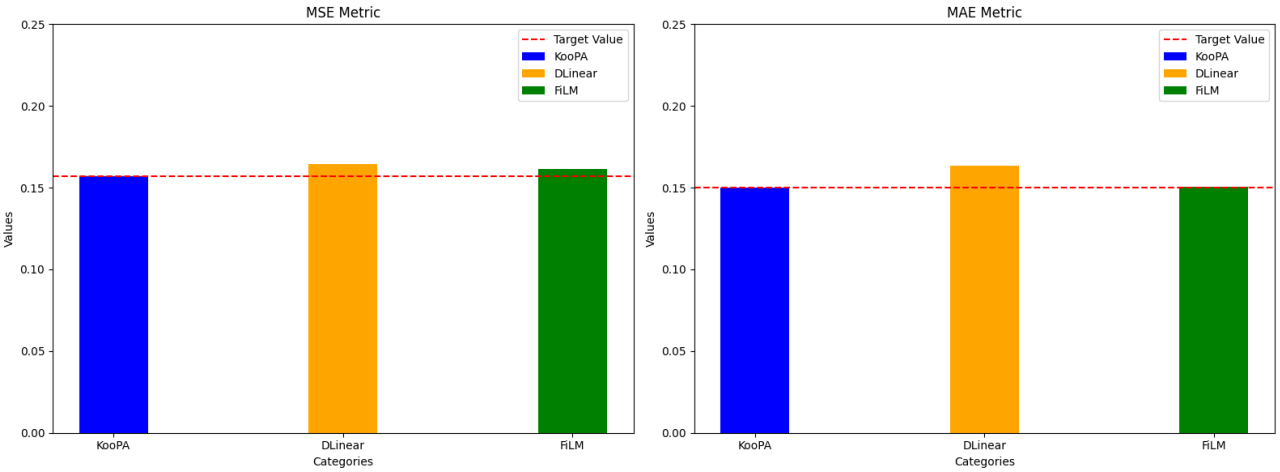


위 MSE / MAE 값을 그래프로 그려낸 결과이다.



2) 결과 분석

KooPA Model과 DLinear Model / FiLM Model을 서로 비교한 결과 근소한 차이지만 MSE / MAE 값이 모두 낮게 나와 예측 성능이 우수한 것을 확인할 수 있었다.



Ⅲ 주요 결과 및 시사점

1. 주요 결과 요약

가. 핵심 결과 정리

1) 가볍지만 강력한 모델

KooPA Model은 타 모델과의 비교를 통해, 시/공간적 효율성과 더불어, MSE/MAE 값 모두가 낮게 출력되는 결과를 확인하였다. 이를 통해, 모든 요소를 효과적으로 균형있게 해결하는 능력을 가진 모델이라는 것을 파악할 수 있었다.

2) 비정상 시계열 데이터에 적용

KooPA 모델은 비정상 시계열 데이터를 처리하는 데에 있어서 뛰어난 성과를 보였습니다. 복잡한 패턴을 캡처하고 다양한 동역학에 대응하여, 비정상 시계열 데이터에 대한 견고한 예측 성능을 입증하였습니다.

나. 결과 값에 대한 특징

1) 정확도와 효율성

Koopa모델은 높은 정확성과 효율성 사이에서 적절한 균형을 달성하여 학습시간을 감소시키고 메모리 소비를 절약한다.

2) 효용성

Koopa의 다양한 동역학에 대한 적용성은 다른 LookBack 창 내에서 예측 범위를 확장하여 높은 효용성을 발휘한다.

2. 결과 활용 및 시사점

가. 비즈니스 및 학문적 활용에 대한 분석

1) 비즈니스적 활용

Koopa 모델은 정확하고 효율적인 시계열 예측이 필요한 금융, 에너지, 물류 등과 같은 산업에 큰 잠재력을 가지고 있다. 비정상적인 데이터에 대한 실시간 의사 결정 지원 시스템에 적합하다.

2) 학문적 활용

학계에서는 Koopa가 비정상적인 시계열 동역학의 이해를 높이는 데 기여하며 Koopman 이론의 통합은 동적 시스템 분석 및 예측 방법론에 대한 추가 연구의 가능성을 열어준다.

나. 보고서의 시사점

1) 시계열 예측의 진보

Koopa의 다변량 동적 모델과의 성공적인 통합은 시계열 예측 분야에서 큰 발전을 나타낸다. 모델의 적응성과 효율성은 연구자와 실무자 모두에게 가치 있는 도구로 작용한다.

2) 이론과 응용 사이의 격차 해소

Koopa가 실제 시나리오에서 Koopman 이론을 적용함으로써 이론적 프레임워크와 실제 유용성 사이의 간극을 줄였다. 이는 모델의 신뢰성을 향상시키는데 뿐만 아니라 동적 시스템 분석의 전반적인 분야에 기여한다.

Ⅳ Appendix

1. 타 모델의 설명

가. D-Linear

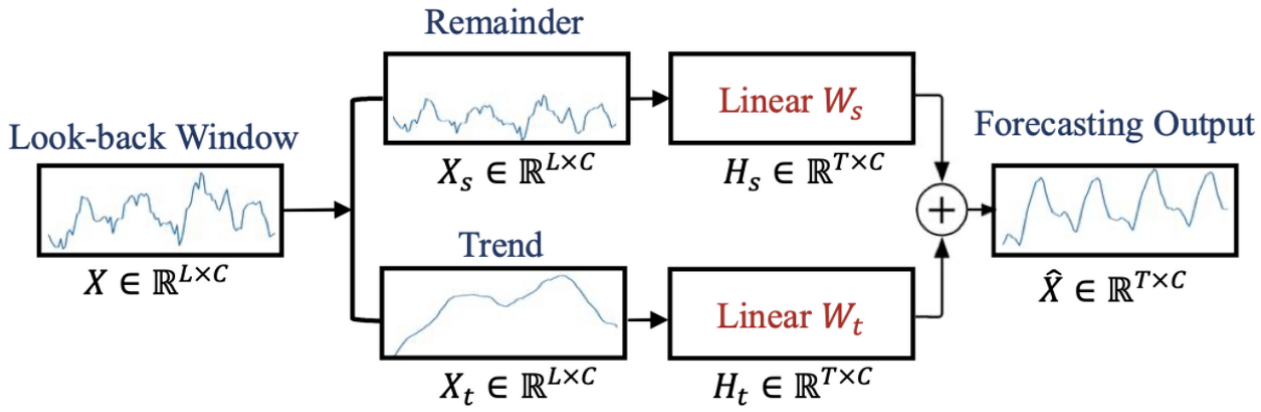
1) D-Linear란?

간단한 선형 모델인 DLinear은 2가지 관점에서 설명 가능하다.

- 1-Layer Linear Network는 미래를 예측하기 위해 과거의 정보를 압축할 수 있는 가장 간단한 모델이다.
- 선행 연구들의 실험을 통해 시계열 분해가 TSF 문제에서 Transformer 기반 모델들의 성능을 높일 수 있는 방법임을 확인할 수 있으며 Model-Agnostic하고 다른 모델에도 쉽게 적용할 수 있다는 점을 선형 모델에 적용 가능하다.

이에 따라 DLinear은 분해 부분과 선형 네트워크로 구성되어 있으며, 이 모델은 시계열 데이터를 Trend Component X_t 와 Remainder Component $X_s = X - X_t$ 로 분해한 이후 두 개의 1-Layer Linear Network를 적용하는 매우 간단한 구조이다.

2) D-Linear의 구조



- O(1) Traversing Path Length : Short Range와 Long Range의 Temporal Relation을 낮은 연산량을 확보할 수 있다.
- Hight Efficiency : Transformer에 비해 사용 메모리와 파라미터가 적기 때문에 연산 속도가 더 빠르다.
- Interpretability : Seasonality와 Trend에 대한 Weight가 분해된 Input에 곱해진 선형 모델이기 때문에 Weight에 대한 분석을 통해 직관적인 해석이 가능하다.
- Easy to use : Transformer와 달리 하이퍼파라미터 튜닝 없이 사용 가능하다다.

2. 참고 자료

가. 참고 문서

1) 논문

1. Liu, Yong, et al. "Koopa: Learning Non-stationary Time Series Dynamics with Koopman Predictors." arXiv preprint arXiv:2305.18803 (2023).
2. Wang, Ming-Chang, Chih-Fong Tsai, and Wei-Chao Lin. "Towards missing electric power data imputation for energy management systems." Expert Systems with Applications 174 (2021): 114743.
3. Brunton, Steven L., et al. "Modern Koopman theory for dynamical systems." arXiv preprint arXiv:2102.12086 (2021).

나. 참고 사이트

1) 블로그

1. LTSF-Linear(DLinear, NLinear) 논문 리뷰/구현 ref) tistroy
2. DLinear(2022)_논문 리뷰 ref) velong.io
3. TimeSeries) Transformer보다 좋다는 LSTF-Linear 알아보기 ref) tistroy
4. 유효전력, 무효전력, 피상전력 ref) blog.naver

2) github 사이트

1. thuml / Time-Series-Library