LG Aimers 3기 코드 발표

온라인 채널 제품 판매량 예측 AI 온라인 해커톤

팀명: kimsh

팀원: 김수형, 홍상호, 서성민, 김효준

목차

01. EDA - 범주형 컬럼

02. EDA - 수치형 컬럼

03. 프로펫을 통한 결측치 보간

04. DATA SMOOTHING

05. 적용 모델 소개

06. 하이퍼파라미터 튜닝

07. 최종 결과

01. EDA – 범주형 컬럼

제품 분류별 그룹화

- 대분류 5, 중분류 11, 소분류 53 개로 확인됩니다.
- 앞 네자리 B002 는 공통번호, 중간 네자리는 대·중·소 분류번호, 마지막 네자리는 분류별 순번입니다.
- 가장 많은 판매량을 보유한 대분류는 B002-C001-0002
- 가장 많은 판매량을 보유한 중분류는 (대분류 B002-C001-0001) , B002-C002-0001
- 가장 많은 판매량을 보유한 대분류와 중분류는 서로 다른 그룹임을 알 수 있습니다.
- 이는 가장 많은 판매량을 보유한 대분류 B002-C001-0002는 가장 많은 중분류 개수인 7개로 구성되어 있기 때문입니다.

대분류	중분류	소분류	개수
B002-C001-0001	B002-C002-0001	B002-C003-0001	751
		B002-C003-0002	663
		B002-C003-0003	1426
		B002-C003-0004	426
.....			
B002-C001-0004	B002-C002-0009	B002-C003-0047	111
B002-C001-0005	B002-C002-0011	B002-C003-0052	392
		B002-C003-0053	221

<Table1. 제품 분류별 그룹화 결과>

02. EDA – 수치형 컬럼

판매량 그래프를 확인

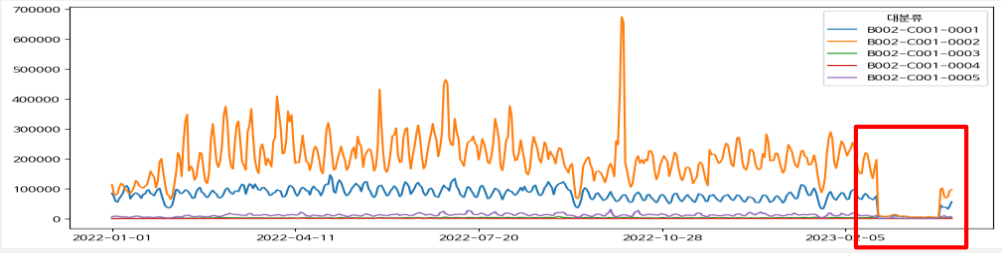
- 대분류별 일자별 판매량 합계 데이터를 추가 생성하여 Line Plot을 통해 판매량 추이를 확인합니다.
- 그 결과 2023-02-23 이후부터 판매량이 급격히 감소하는 것을 알 수 있습니다.

⬇

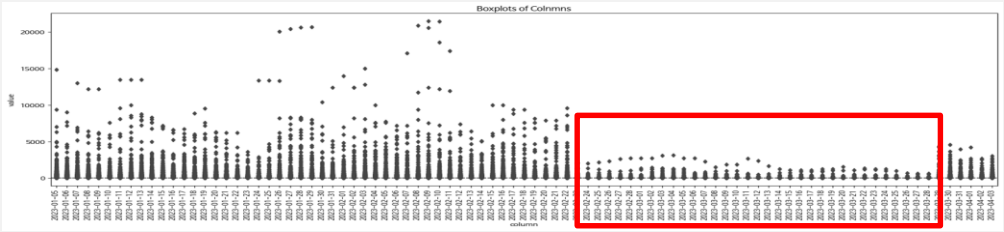
- 판매량 감소 구간을 Box Plot 으로 더 자세히 확인합니다.
- 2023-02-23 ~ 2023-03-28 기간에 판매량이 현저히 적은 것을 확인 할 수 있습니다.

⬇

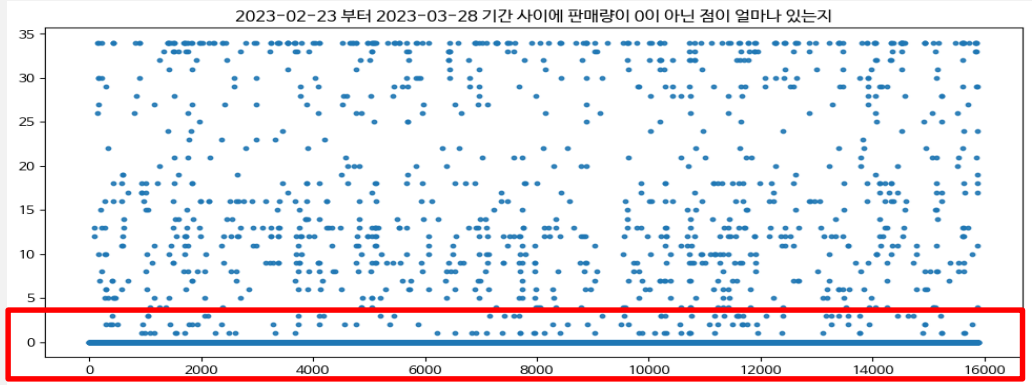
- '2023-02-23 ~ 2023-03-28' 기간 동안 판매량이 0이 아닌, 실제 판매가 발생한 기간이 얼마인지 산점도로 표현합니다. (y축은 판매 발생수, x 축은 제품ID)
- 판매 발생수가 1이상에 타점된 부분은 전체중 일부분에 불과한 것을 확인 됩니다.
- '0 값'들이 대부분을 차지하는 결측구간(2023-02-23 ~)을 보간할 필요성이 있습니다.
- **따라서 이전값(~ 2023-02-23)을 통해 결측구간(2023-02-23 ~)을 예측하여 보간을 실시하고 보간이 완료된 데이터로 미래의 21일치를 최종적으로 예측하는 단계를 진행할 것입니다.**



<Figure 1. 대분류별 일자별 판매량 합계의 Line Plot>



<Figure 2. 2023-02-23 ~2023-03-28 기간 일자별 판매량 Box Plot>



<Figure 3. 2023-02-23 ~2023-03-28 기간 판매 발생수 Scatter Plot>

03. 프로펫을 통한 결측치 보간

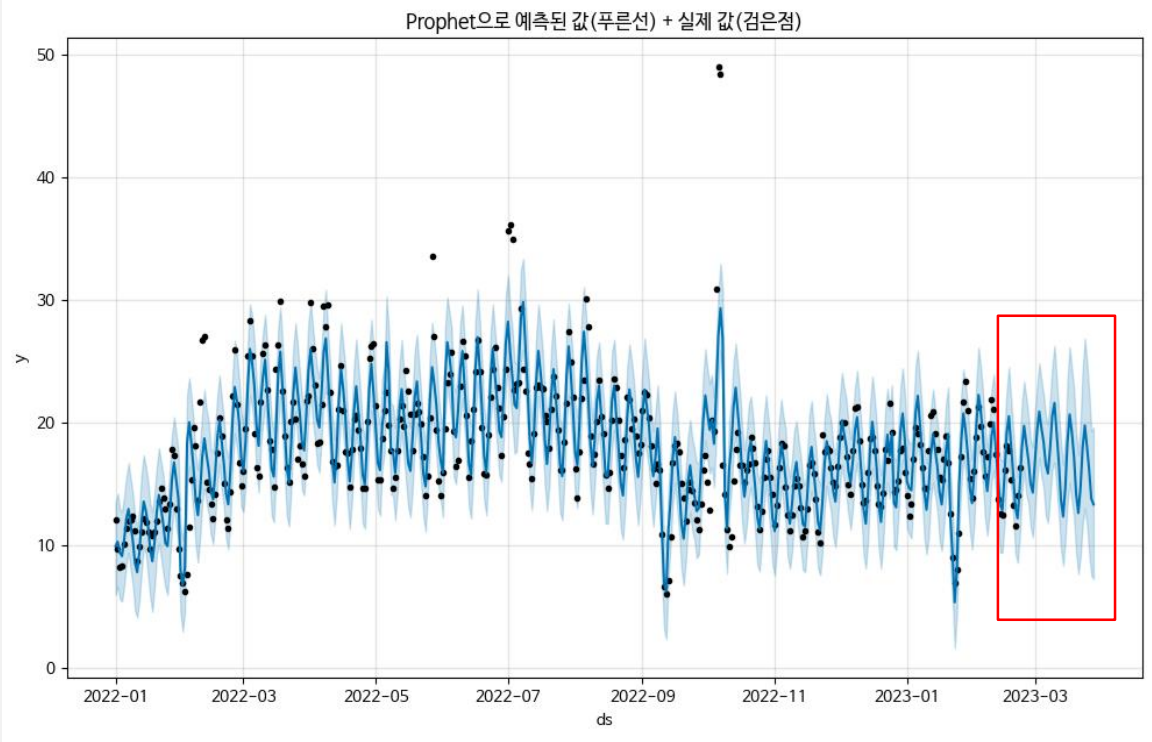
프로펫 소개

- Prophet은 페이스북에서 개발한 시계열 예측 라이브러리로써 이론을 잘 모르더라도 그래프의 개형만 보고도 손쉽게 예측을 할 수 있게 해주는 모델입니다.

- 또한, 계절성의 타입도 연, 월, 주 등등 원하는 대로 적용할 수 있고 비선형적 추세가 있어도 문제없이 잘 작동합니다.

- 계절적 효과가 강할 때 가장 큰 효과를 발휘한다고 생각하는데, 저희의 데이터는 계절성이 굉장히 강하다고 생각해서 (7의 배수 단위) 해당 모델을 사용해서 중간에 빈 34일간의 데이터를 대체하는 곳에 이용하기로 했습니다.

- 이전의 추세와 계절성을 가장 그대로 받아들인 예측치가 이후에 진행될 21일동안 예측 값을 만들 때 가장 효과가 좋을 것이라고 판단하여서 이런 결정을 내렸습니다.



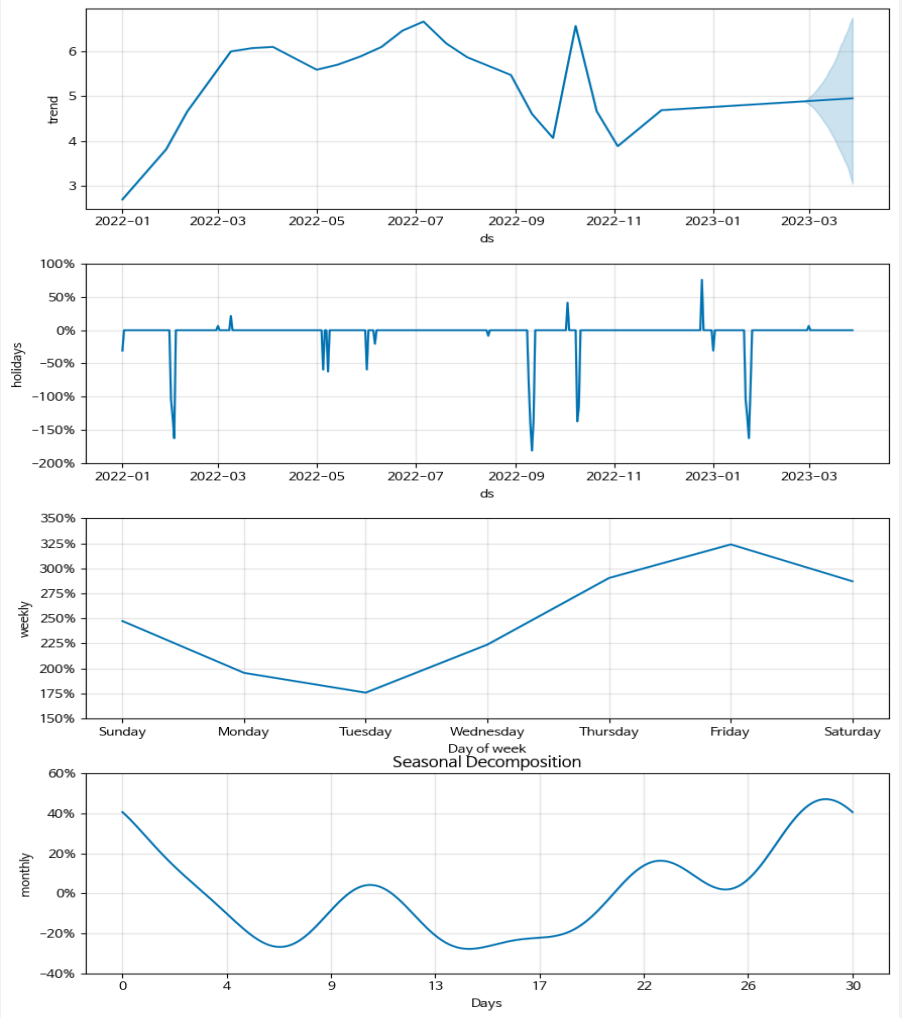
<Figure 4. Prophet 을 통한 34일 간 예측치>

- 위 그래프는 각 날마다의 판매량을 모두 더한 점으로 Prophet 모형을 이용해 34일간 예측치를 얻은 그래프입니다.

03. 프로펫을 통한 결측치 보간

프로펫 Seasonal Decomposition

- Prophet모형을 이용하면 Seasonal Decomposition도 굉장히 간단하게 진행할 수 있습니다.
- 오른쪽의 그래프는 순서대로 Trend, Holiday, Weekly Seasonality, Monthly Seasonality 입니다. (Holiday라는 개념이 생소한데 추후에 설명하겠습니다.)
- 그래프를 확인하면 확실히 요일마다 판매량이 차이가 있음을 확인할 수 있습니다.
- Weekly Seasonality(7), Monthly Seasonality(30) 가 확실히 잘 나타나는 것을 알 수 있습니다.



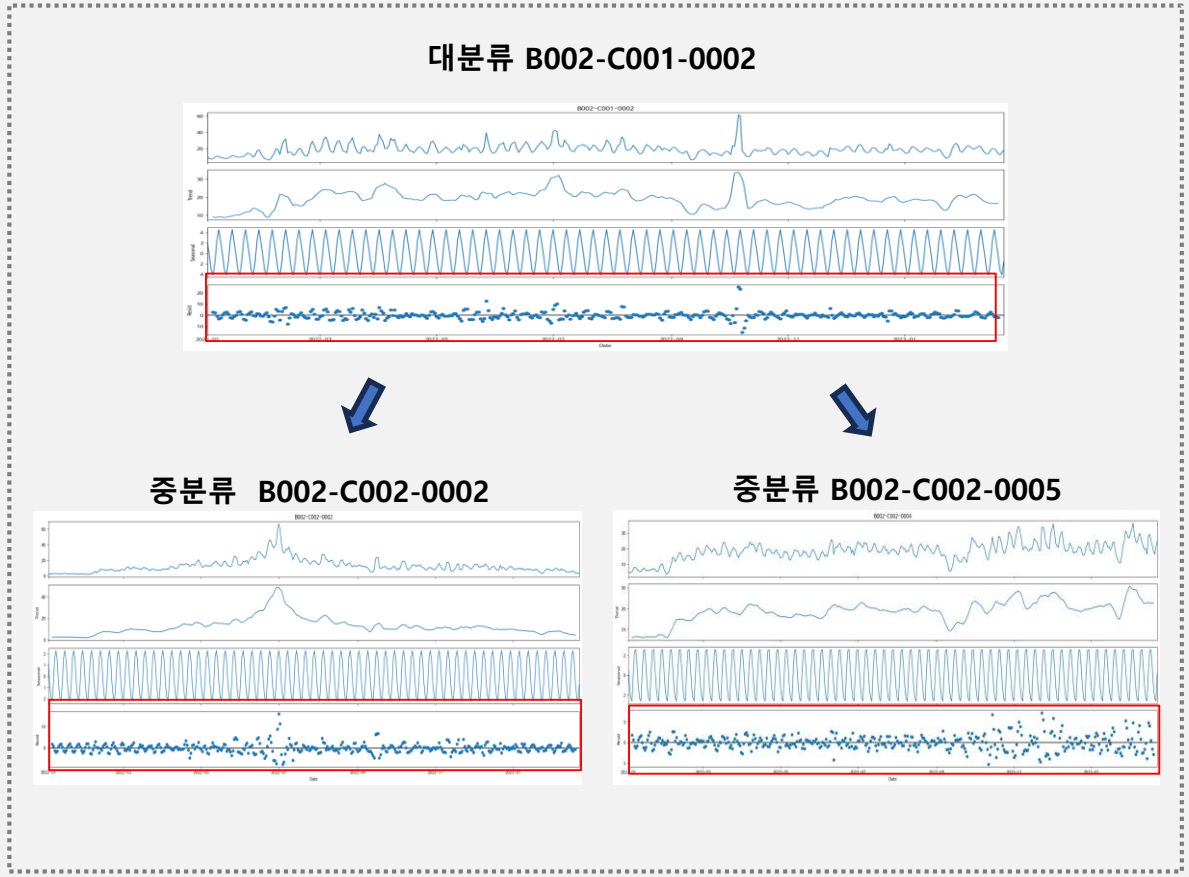
<Figure 5. Prophet 을 통한 Seasonal Decomposition >

03. 프로펫을 통한 결측치 보간

모든 대분류와 중분류의 주기성 확인

- 먼저 Weekly Seasonality, 즉 주기(Period)를 7로 하고 대분류별 Seasonal Decompose 를 합니다.
- 그 결과 5개 대분류 중 2개에서 잔차 그래프가 패턴을 보입니다.
- 잔차 그래프가 패턴을 보인다는 주기 7이 적절하지 않다고도 할 수 있습니다.

- 중분류별 계절 분해를 진행합니다.
- 잔차 그래프가 패턴을 보였던 대분류 인 'B002-C001-0002 ' 에 속하는 중분류 'B002-C002-0002 ' 와 같이 잔차 그래프가 트렌드를 가지는 것도 있지만 'B002-C002-0005' 와 같이 잔차 그래프가 트렌드를 가지지 않는 중분류가 대부분입니다.
- 따라서 주기 7은 적절한 것으로 판단됩니다.
- 추후 분석모델에 주기는 7 단위로 활용할 수 있습니다.



<Figure 6. 대중분류별 Seasonal Decomposition >

03. 프로펫을 통한 결측치 보간

프로펫 파라미터 결정 – 경향과 계절성을 더 유연하게

이는 이미 시행착오 끝에 지정한 하이퍼 파라미터입니다.

```
from prophet import Prophet
m = Prophet(
    # trend
    changepoint_prior_scale=0.5, # 경향을 더 유연하게 받아들임.
    # seasonality
    seasonality_prior_scale = 0.5, # 계절성도 더 유연하게 받아들임.
    weekly_seasonality = 20,
    daily_seasonality=False,
    yearly_seasonality=False,
    seasonality_mode='multiplicative' # 계절성의 크기를 가변형으로 잡고
)
m.add_seasonality(name='monthly', period=30.5, fourier_order=5) # monthly 계절성 추가
m.add_country_holidays(country_name='KR') # 한국의 휴일 날짜를 자동으로 입력해줌
m.fit(series)

future = m.make_future_dataframe(periods = 34)
forecast = m.predict(future)

fig1 = m.plot(forecast)
plt.title('Prophet으로 예측된 값(푸른선) + 실제 값(검은점)')
fig2 = m.plot_components(forecast) # Seasonal Decomposition
plt.show()
# 약간씩 벗어나는 점도 있고, 굉장히 변동이 큰 경우에는 놓치는 점도 있지만, 대체로 잘
```

<Figure 7. Prophet 구동 코드>

- 저희는 모델을 선정할 때 경향과 계절성을 더 유연하게 받아들이도록 설정하였습니다. (0.05 -> 0.5)
- 이를 통해 앞선 값들을 더 잘 fitting 하게 되면서 overfitting 에 대한 가능성이 증가 할 수 있지만 다음과 같은 이유로 이점이 더 크다고 생각했습니다.
- 단순히 동일 데이터 내에서 결측을 대체할 때만 사용할 모델이기 때문에 overfitting은 크게 고려하지 않았습니다.
- 또한 판매량들의 경향, 계절성을 정확하게 잡아내서 그 흐름에 따른 미래를 예측 해주는 게 더 합리적이라고 판단했습니다.

03. 프로펫을 통한 결측치 보간

프로펫 파라미터 결정 – 한국의 공휴일 추가

이는 이미 시행착오 끝에 지정한 하이퍼 파라미터입니다.

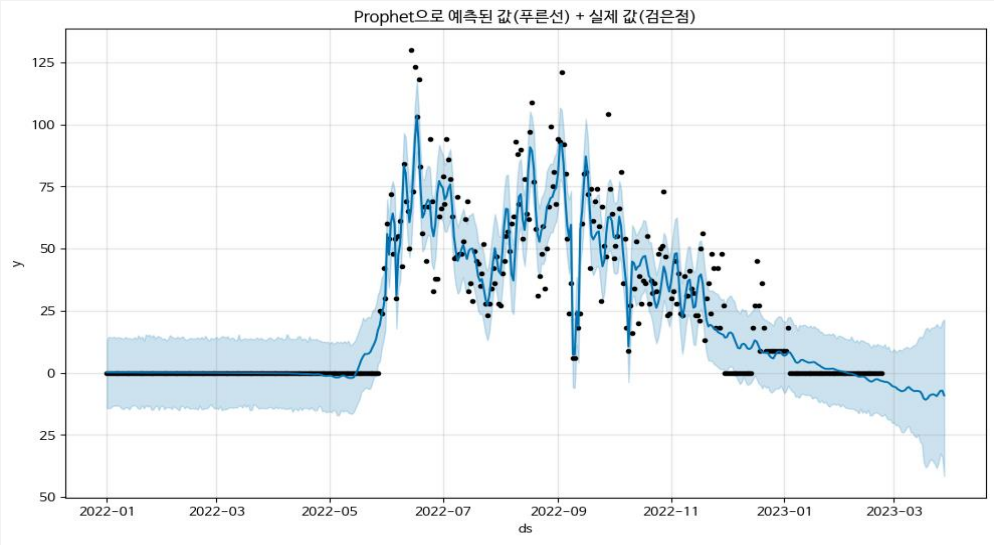
```
from prophet import Prophet
m = Prophet(
    # trend
    changepoint_prior_scale=0.5, # 경향을 더 유연하게 받아들임.
    # seasonality
    seasonality_prior_scale = 0.5, # 계절성도 더 유연하게 받아들임.
    weekly_seasonality = 20,
    daily_seasonality=False,
    yearly_seasonality=False,
    seasonality_mode='multiplicative' # 계절성의 크기를 가변형으로 잡고
)
m.add_seasonality(name='monthly', period=30.5, fourier_order=5) # monthly 계절성 추가
m.add_country_holidays(country_name='KR') # 한국의 휴일 날짜를 자동으로 입력해줌
m.fit(series)

future = m.make_future_dataframe(periods = 34)
forecast = m.predict(future)

fig1 = m.plot(forecast)
plt.title('Prophet으로 예측된 값(푸른선) + 실제 값(검은점)')
fig2 = m.plot_components(forecast) # Seasonal Decomposition
plt.show()
# 약간씩 벗어나는 점도 있고, 굉장히 변동이 큰 경우에는 놓치는 점도 있지만, 대체로 잘
```

<Figure 8. Prophet 구동 코드>

- m.add_country_holidays(country_name='KR')은 모델에 holiday를 추가해주는 부분입니다. 해당 코드를 이용하면 한국의 공휴일을 자동으로 'holiday'로 추가해줍니다.

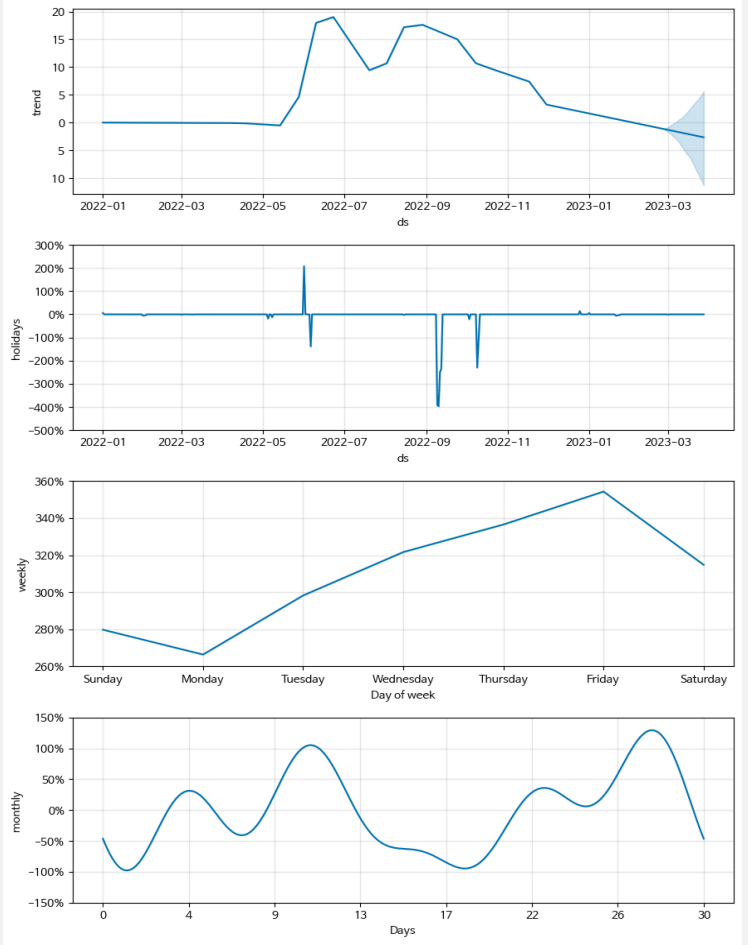
- Holiday로 지정된 날은 이후 예측선을 그리는 데에 영향을 극단적으로 줄입니다. 이는 해당 날짜들은 이상치가 존재하는 날이라고 모델에게 알려주어 예측의 성능을 높일 수 있습니다.

03. 프로펫을 통한 결측치 보간



<Figure 9. Prophet 을 통한 결측치 예측 그래프>

제품 ID = 7 만 추출 후
Seasonal Decomposition 확인



<Figure 10. 제품 ID = 7 인 제품에 대한 Seasonal Decomposition>

- 결측구간 예측 결과 프로펫 모델은 위처럼 음수값을 예측하는 경우도 있었는데, 저희는 최근의 동향을 반영하는 게 중요하다고 생각해서 하한, 상한을 두지 않고 음수 값을 모두 0으로 치환했습니다.
- 예를 들어서 판매가 존재하다가 갑자기 최근 들어서 판매가 발생하지 않았다면, 그 이후에도 판매가 발생하지 않겠다는 예측이 합리적이라 생각하였습니다.
- 그렇지 않고 상한과 하한을 지정하면 이런 최근 흐름을 놓치는 경우가 종종 생겨서 더 단순하고 강경한 방법을 채택하였습니다.

04. DATA SMOOTHING

Savitzky-Golay filter

- 논문 리뷰를 통해 BASELINE의 모델인 LSTM 알고리즘을 돌리는데 있어서 평활기법을 적용하는 것이 예측 성능을 올리는 데 도움이 된다는 것을 알 수 있었습니다.
- 위 논문에서는 Savitzky-Golay, 지수 평활법, 가중치 이동 평균 평활 법을 LSTM 알고리즘에 적용하고 결과를 비교하였습니다.
- 결과는 다음과 같았고 결과적으로 Savitzky-Golay 평활법을 적용한 LSTM 알고리즘이 예측 성능에 유의미한 좋은 결과를 보였습니다.

<Figure 11. 평활법별 손실률 그래프>

Table 2. Comparison of Learning Loss and Validation Loss According to Application of LSTM and Smoothing Technique

Model&Filter	Train Loss	Validation Loss
LSTM	2.085	0.00026
Savitzky-Golay : LSTM	1.906	0.00012
Exponential Smoothing : LSTM	8.464	0.00063
Weighted Moving Average : LSTM	5.477	0.00054

Table 3. Comparison of learning loss and validation loss according to application of LSTM/GRU and smoothing technique(2020 times) : 365 days

Model&Filter	Train Loss	Validation Loss
LSTM	3.0556	0.00005
GRU	3.4426	0.00003
Savitzky-Golay LSTM	1.4659	0.00002
Savitzky-Golay GRU	4.7761	0.00003
Exponential Smoothing : LSTM	7.7591	0.00009
Exponential Smoothing : GRU	8.8429	0.00007
Weighted Moving Average : LSTM	7.4146	0.00006
Weighted Moving Average : GRU	6.0544	0.00006

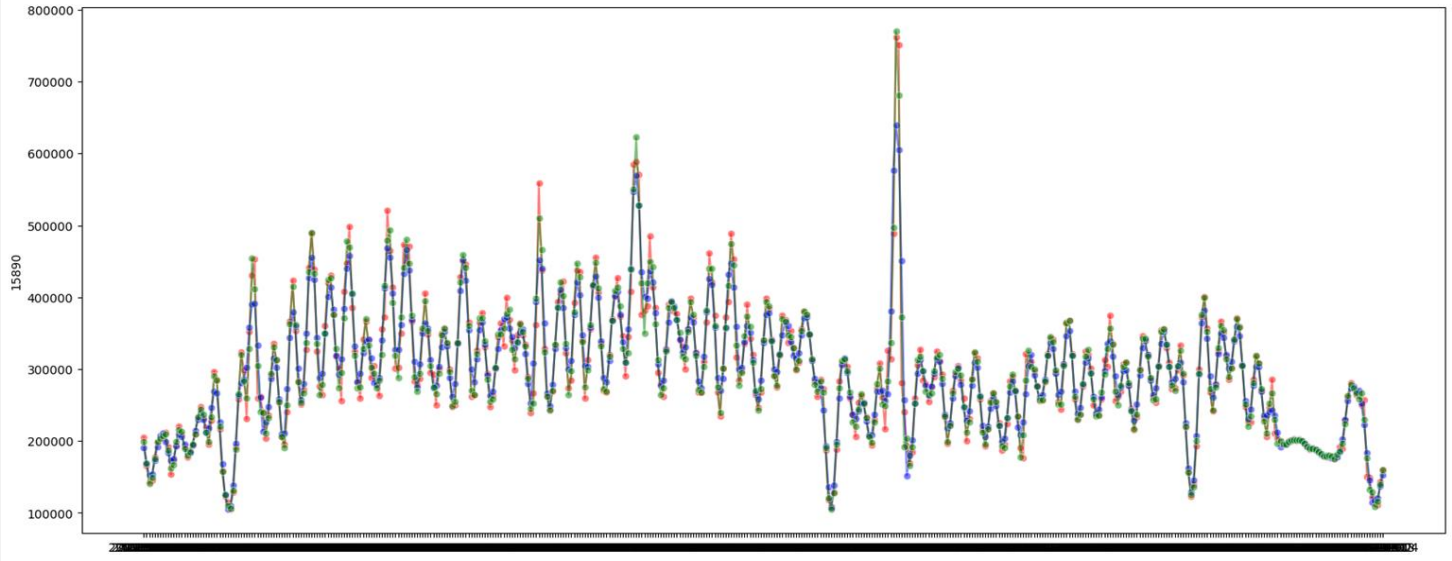
박태진, 심갑식. "LSTM 학습모델의 성능 향상을 위한 디지털 신호 필터의 비교연구: 학습 모델 예측 성능 향상에 영향을 미치는 비트코인 시계열 데이터 전처리 및 스무딩 기법 적용." 한국컴퓨터정보학회논문지 (컴퓨터정보논문지) 28, no. 1 (2023): 17-26. DOI: 10.9708/jksci.2023.28.01.017.

04. DATA SMOOTHING

Savitzky-Golay filter 적용

- Savitzky-Golay 평활은 `scipy`의 `signal` 모듈에서 `savgol_filter` 함수를 이용하여 구현했습니다.
- 전처리 부분에서 7일 주기를 확인하였으므로 window_length = 7로 고정했습니다
- 차수(polyorder)부분은 너무 커지면 오류를 줄일 수 있지만 과적합이 발생할 수 있으므로 3차, 5차로 설정하여 2개를 비교하였고 mode = "nearest"로 설정하여 가장 가까운 데이터 포인트를 사용하여 스무딩을 진행하였습니다.

```
td_smooth_3 = signal.savgol_filter(train2, window_length = 7, polyorder = 3, mode="nearest")
td_smooth_5 = signal.savgol_filter(train2, window_length = 7, polyorder = 5, mode="nearest")
td_smooth_3
```



<Figure 12. Savitzky-Golay filter 적용 그래프>

- 빨강 : 원본
- 초록 : polyorder = 5 > smoothing 효과가 미비
- 파랑 : polyorder = 3 > 적용

04. DATA SMOOTHING

Savitzky-Golay filter 성능확인

- PROPHET으로 결측치 대체한 데이터 + MINMAX SCALING + BASELINE LSTM

888258	last_year_replace_baseline.csv 결측치 대체v1o / minmax edit	2023-08-12 17:24:02	0.5024993414 -

- PROPHET으로 결측치 대체한 데이터 + Savitzky-Golay SMOOTHING + MINMAX SCALING + BASELINE LSTM

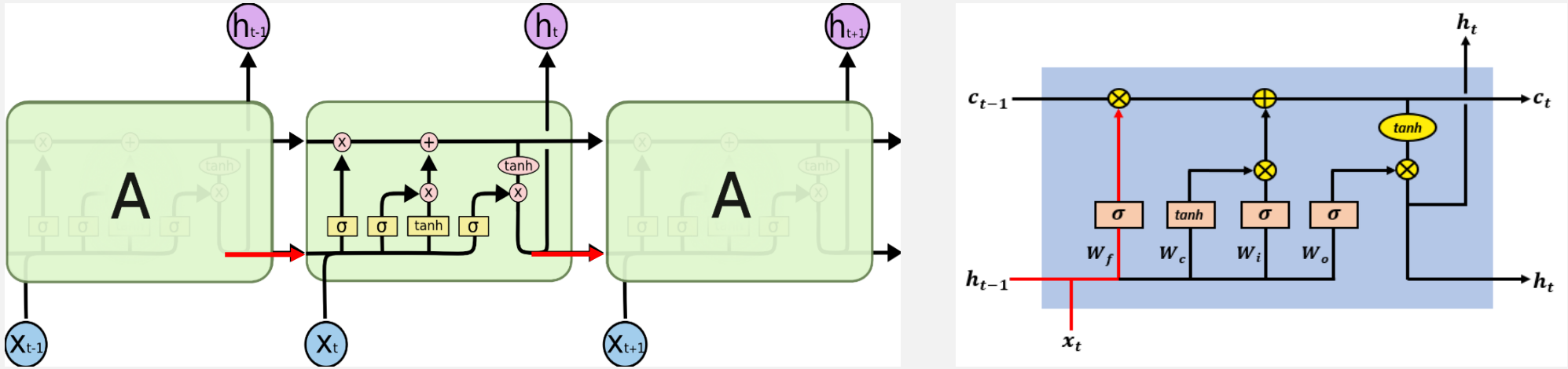
887570	submit_LSTM_smooth_minmax.csv 결측치 v1/ savgol_ smoothing/ minmax /round edit	2023-08-11 16:15:24	0.521126062 -

- 약 0.02 정도 큰 점수를 얻게 되었습니다.
- 판매량 데이터에도 Savitzky-Golay 평활법을 적용한 LSTM 알고리즘이 예측 성능에 유의미한 결과를 가져다줌을 확인했습니다.

05. 적용 모델 소개

LSTM (Long Short Term Memory)

- 기존의 RNN 모델을 개선하여 출력과 먼 위치에 있는 정보를 기억하게 하게 하여 긴 의존 시간을 필요로 하는 학습을 수행합니다.



< Figure13. How LSTM networks solve the problem of vanishing gradients, Nir Arbel >

05. 적용 모델 소개

Bi-LSTM (Bidirectional LSTM)

- 논문 리뷰를 통해 양방향으로 데이터를 처리하는 접근 방식이 시계열 예측의 정확성을 평균 37.78% 향상시킬 수 있음을 알 수 있었습니다.
- 참고한 논문은 ARIMA, 단방향 LSTM (Long Short-Term Memory) 및 양방향 LSTM (BiLSTM) 모델의 성능, 정확성 및 행동 훈련을 분석하고 비교한 실험 결과를 정리했습니다.
- 실험에서 다루는 주요 연구는 순방향 (Forward)과 역방향 (Backward)의 데이터 훈련을 결합하여 시계열 예측의 정밀도에 긍정적이고 유의한 영향을 미칠 수 있는지에 초점을 맞췄습니다.
- 따라서 저희는 순방향과 역방향의 정보를 모두 활용하여 시퀀스 데이터를 처리하는 Bi-LSTM(Bidirectional Long Short-Term Memory) 모델을 적용했습니다.

TABLE II: RMSEs of ARIMA, LSTM, and BiLSTM models.

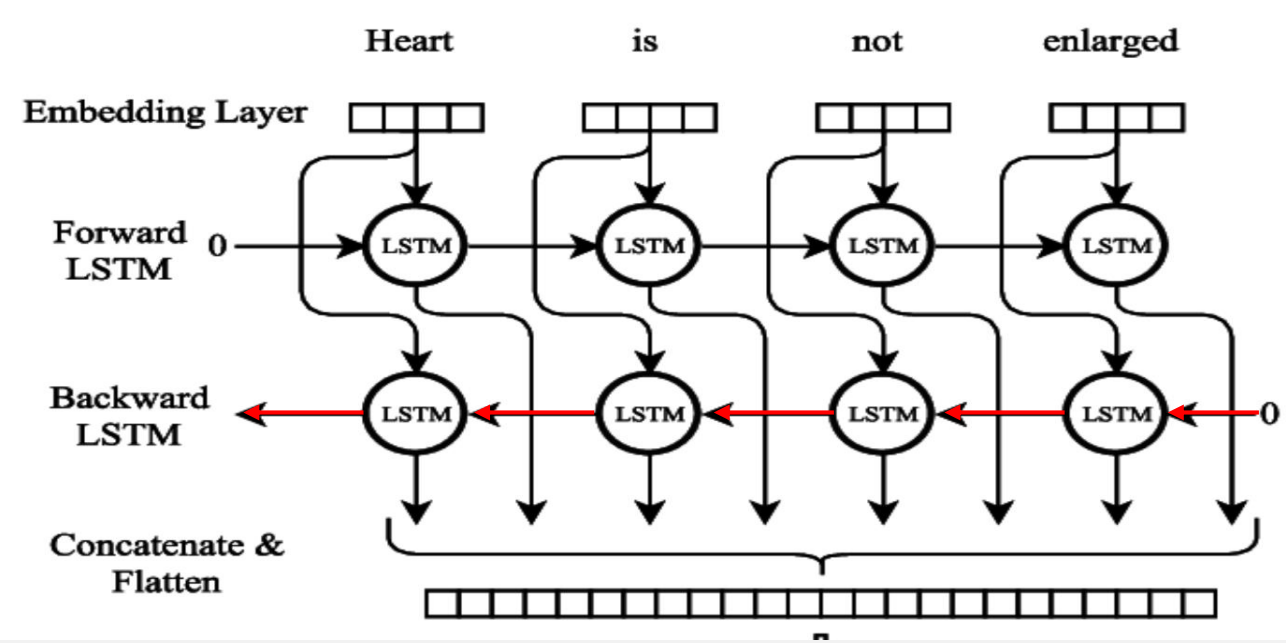
Stock	RMSE			% Reduction		
	ARIMA [20]	LSTM	Bi LSTM	BiLSTM over LSTM	BiLSTM over ARIMA	LSTM over ARIMA
N225monthly	766.45	102.49	23.13	-77.43	-96.98	-86.66
IXIC.daily	34.61	2.01	1.75	-12.93	-94.94	-94.19
IXIC.weekly	72.53	7.95	11.53	45.03	-84.10	-89.03
IXIC.monthly	135.60	27.05	8.49	-68.61	-93.37	-80.00
HSI.monthly	1,306.95	172.58	121.71	-29.47	-90.68	-86.79
GSPC.daily	14.83	1.74	0.62	-64.36	-95.81	-88.26
GSPC.monthly	55.30	5.74	4.63	-19.33	-91.62	-89.62
DJI.daily	139.85	14.11	3.16	-77.60	-97.77	-89.91
DJI.weekly	287.60	26.61	23.05	-13.37	-91.98	-90.74
DJI.monthly	516.97	69.53	23.69	-65.59	-95.41	-86.50
IBM.daily	1.70	0.22	0.15	-31.18	-91.11	-87.05
Average	302.96	39.09	20.17	-37.78	-93.11	-88.07

Siami Namini, S., Tavakoli, N., & Siami Namin, A. (2019). The Performance of LSTM and BiLSTM in Forecasting Time Series. In 2019 IEEE International Conference on Big Data (Big Data). DOI: 10.1109/BigData47090.2019.9005997.

05. 적용 모델 소개

Bi-LSTM (Bidirectional LSTM)

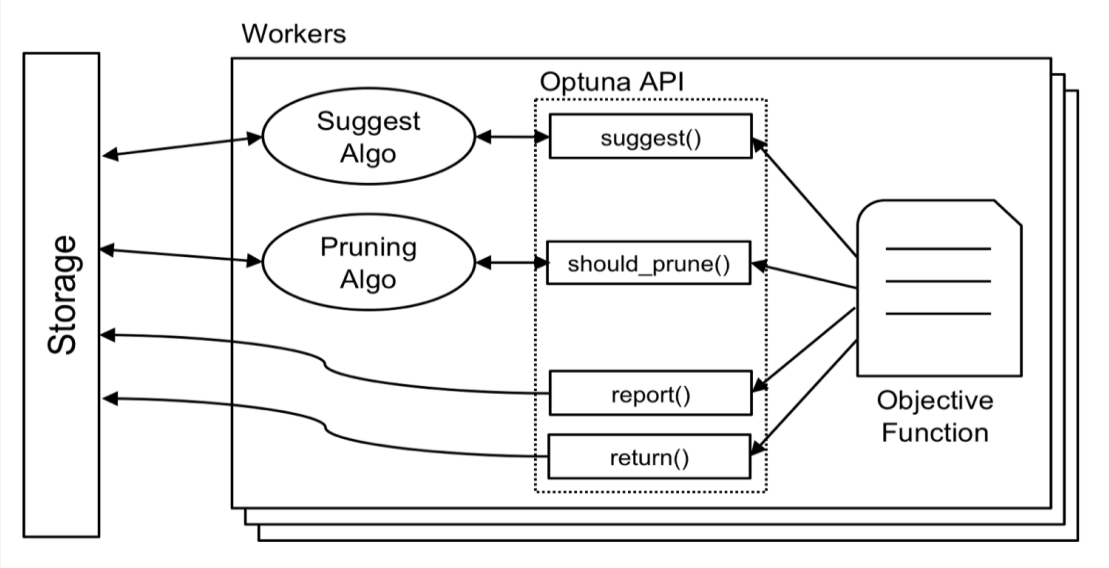
- 양방향 LSTM은 정방향으로만 학습을 진행하는 대신 마지막 노드에서 역방향으로 실행되는 LSTM을 추가하여 시계열 또는 시퀀스 데이터의 시간 스텝 간의 양방향 장기 종속성을 학습합니다.



<Figure14. Modelling Radiological Language with Bidirectional Long Short-Term Memory Networks, Cornegruta et al.>

Optuna

- HyperParameter는Learning Rate나 Hidden Layer 개수 등으로 튜닝을 통해 최적해를 추적합니다.
- Optun는 Grid, Random Search를 포함하여 효율적인 Tuning을 위한 SOTA 알고리즘을 제공합니다.



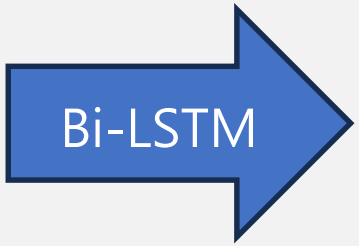
<Figure15. Optuna: A Next-generation Hyperparameter Optimization Framework, Takuya Akiba et al.>

07. 최종 결과

최적화된 하이퍼파라미터

Hyper - parameter	
TRAIN WINDOW SIZE	91
PREDICT SIZE	21
EPOCHS	10
LEARNING RATE	9.90E-05
BATCH SIZE	4096
DROP OUT	1.10E-01
SEED	41

<Table 2. 최적화된 파라미터 값>



최종 Score

Score	
Public	Private
0.56976	0.54956

<Table 3. 최종 score>

감사합니다.