LGaimers 7기 온라인 해커톤

팀명: 망고주스벤티

팀원: 조정원, 김송목, 변수현, 김민서, 박준식

Feature

EDA 및 데이터 전처리 방향성

1. 데이터 및 칼럼의 부족

   : 시계열 데이터임에도 불구하고 train 데이터가 1년 반 정도의 데이터만 있었고, 칼럼 또한 영업일자, 영업장명_메뉴명, 매출수량 이라는 3개의 칼럼만 존재

   ➔ 여러 개의 feature 생성 후 중요 feature 중심으로 feature 고도화
   ➔ 임의로 추세 반영 가능한 feature 고려

2. 매출수량 == 0인 날이 50% 이상

   : 매출수량이 0인 row가 50% 이상이었고 몇 메뉴는 7-80% 이상이 0인 경우도 존재

   ➔ 매출수량이 0이라는 점에 집중한 feature 생성
   ➔ 매출수량이 0이 아닌 날의 data에 집중한 feature 생성

생성한 feature 정리

기본 date 변수	year	기존 데이터 영업일자를 정제하여 생성
	month	기존 데이터 영업일자를 정제하여 생성
	day	기존 데이터 영업일자를 정제하여 생성
	weekday	월요일부터 일요일까지 0부터 6으로 범주화하여 생성
	is_holiday	공휴일을 0과 1로 이진분류하여 생성
	is_weekend	주말을 0과 1로 이진분류 하여 생성
	is_sandwich	해당 날짜를 기준으로 하루 전과 하루 후가 모두 휴일,공휴일인 경우 1 아닌경우 0 으로 이진분류하여 생성
	is_before_holiday	휴일 전인 경우 1, 그 외 경우 0으로 분류하여 생성
	is_after_holiday	휴일 후인 경우 1, 그 외 경우 0으로 분류하여 생성
menu 관련 변수	menu_rank	(메뉴의 순위 - 1) / (업장 내 메뉴개수 -1)
	menu_category	menu별 category를 0부터 6까지 7가지 경우로 범주화 하여 생성
기간 분할 변수	quarter	분기별로 0부터 3까지 4가지 경우로 범주화 하여 생성
	season	계절별로 0부터 3까지 4가지 경우로 범주화 하여 생성
	solar_term	해당 날짜가 속한 절기를 0부터 23까지 24가지로 범주화하여 생성
	quarter_sum	메뉴별 분기 단위 누적 판매량
	season_sum	메뉴별 계절 단위 누적 판매량
	solar_term_sum	메뉴별 분기 단위 누적 판매량

생성한 feature 정리

date 특성 활용	weekday_score	메뉴별 요일 가중치(각 요일 0~1 사이 값)
	month_score	메뉴별 월별 가중치(각 월별 0~1 사이 값)
	date_weight	영업장별 개별 날짜 가중치 x 월 가중치
	menu_date_weight	메뉴별 개별 날짜 가중치 x 월 가중치
매출수량 == 0인 경우가 많음	zero_sales_day_ratio	메뉴별 판매가 0인 날 비율
	avg_sales_nonzero_days	메뉴별 매출 수량 / 팔린 날 일수
	avg_sales_nonzero_monthly	메뉴별 매출수량 / 팔린 날 일수 → 월별 평균
	avg_sales_nonzero_seasonly	메뉴별 매출수량 / 팔린 날 일수 → 계절별 평균
	avg_sales_nonzero_weekday	메뉴별 매출수량 / 팔린 날 일수 → 요일별 평균
	var_sales_nonzero_days	메뉴별 매출 수량 / 팔린 날 일수
수요 변동성	demand_volatility	매출수량 변동성
매출 평균	avg_sales_all_days	메뉴별 매출수량 / 전체 일수

modeling

앙상블 전략(Late Fusion)

A 파이프라인(XGBoost +LightGBM +CNN-LSTM)

- 시계열 기반 피처들을 활용하여 트리 모델 학습
- CNN-LSTM 기반 모델은 최근 35일 윈도우 기반 시계열 구조 학습
- 최종적으로 XGBoost와 LightGBM결과를 가중 평균

B 파이프라인(FT-Dozer)

- 범주형 임베딩 + Dozer Attention 구조를 활용한 PyTorch 기반 시계열 딥러닝 모델
- Attention을 통해 최근 시점에 더 높은 가중치 부여
- 카테고리 특성과 시계열 특성을 동시에 반영 가능

모델 선택 이유

XGBoost/LGBM

데이터셋이 다음과 같은 특성을 지님

- 다양한 수치형, 범주형 변수 존재
- 결측치 많음/희소성 높은 카테고리 존재
- 시계열적 패턴

=> 이러한 세 가지 특성은 트리 기반 부스팅 모델이 강점을 가지는 구조

FT-Dozer

범주형 feature 의 누락을 방지하고 트리 모델의 성능을 보강하기 위해 사용

1. 범주형 feature의 누락 방지

연속값(판매량 시퀀스)과 **범주형 캘린더/메뉴 정보**를 **동시에 임베딩**해 **주의(attention)** 로 조합

2. 최종 모델의 성능 보강

트리모델과 **late-fusion**으로 성능 보강.

CNN

단기 + 장기 패턴을 동시에 학습할 수 있는 모델 필요*[1]

데이터

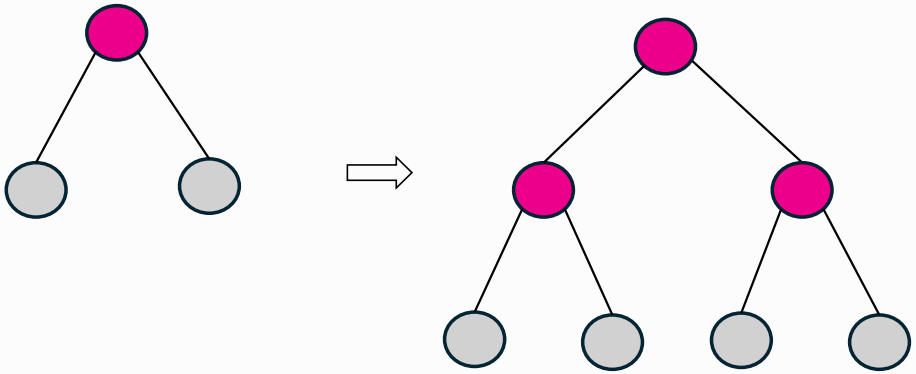
→ 식음업장_메뉴 일별 매출 시계열 데이터

특징

1. 짧은 구간에서의 급격한 변동 (주말·공휴일 급등, 0 매출일 다수)
2. 장기적인 주기성·계절성 (계절/분기/절기 요인)

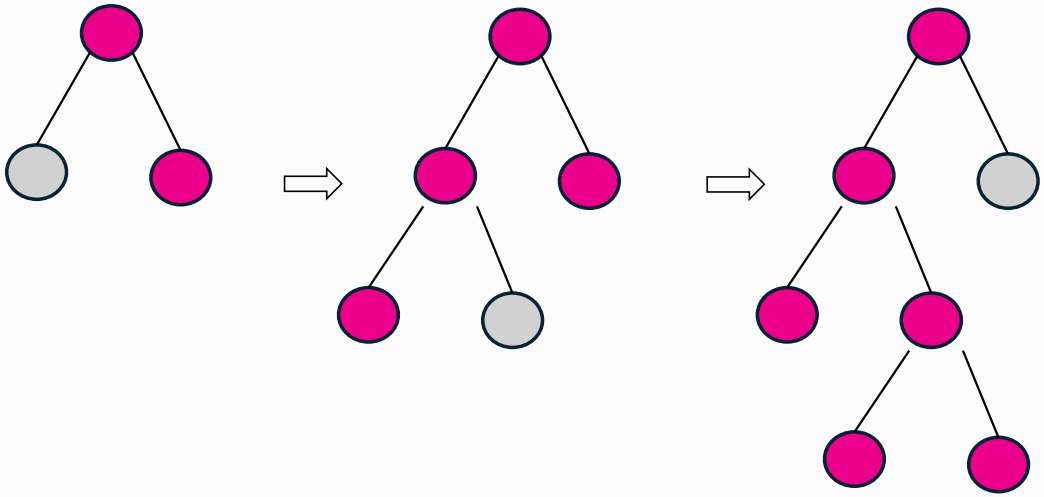
XGBoost/LGBM

XGBoost



오차 보정에 최적화된 Level-wise 성장 방식을 가진
고정밀도 부스팅 모델로, 다양한 문제에서 높은 예측 성능을 보장

LightGBM



빠르고 효율적인 Leaf-wise 성장 구조를 갖춘
고성능 부스팅 트리 모델로, 대용량 데이터 처리와 빠른 실험에 최적화

XGBoost는 '수평'으로 성장하며, 정밀하고 안정적인 예측에 강점
LightGBM은 '수직'으로 성장하며, 속도와 자원 효율성에 최적화
⇒ 두 모델의 트리 성장 방식 차이를 통해 앙상블 시 상호 보완 가능

TF-Dozer

algorithm

1. **범주 임베딩**: 각 범주의 카디널리티에 맞춰 임베딩(4~16차원) 후 concat
2. **DozerAttention(거리 감쇠 + 플래그 게이팅)**

$$Attn\ (i,j) = softmax\ (\frac{Q_j K_j^T}{\sqrt{d_k}} - \lambda|i-j|)$$

λ>0: **가까운 시점**에 가중치를 더 주는 **거리 감쇠**

플래그 게이팅: 공휴일·주말·샌드위치·전/익일 플래그 5개를 R^D로 투영해 **값벡터 V를 시점별 게이트**(0~1) → 이벤트가 있을 때 과거 패턴 참조를 선택적으로 증폭/감쇠

TF-Dozer

앙상블: A파이프라인=트리+시계열 DL, B파이프라인=FT-Dozer, 이후 Late-fusion

1. 테스트 7일 **롤아웃**: 하루씩 예측 → 윈도우 뒤에 **예측값을 이어붙여** 다음 날 예측

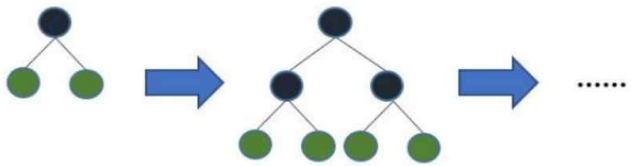
2. FT-Dozer의 1-step 예측 $\hat{y}_B$와 A파이프라인(XGB/LGB + CNN)의 예측 $\hat{y}_A$ 를 검증으로 찾은 가중치 w로 **late-fusion**:

$$\hat{y} = w\hat{y}_A + (1-w)\hat{y}_B$$

3. 이후(최종 단계) **ZI 보정/캡/최소1** 규칙 적용 → 극단치·제로폭주 제어

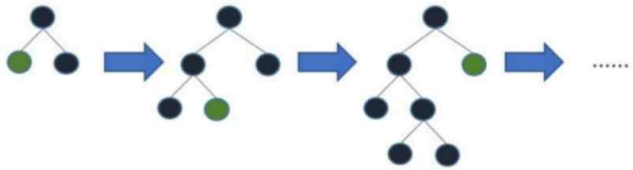
Why CNN-LSTM?

XGBoost:



Level-wise tree growth

LightGBM:



Leaf-wise tree growth

기존 트리 모델의 한계점

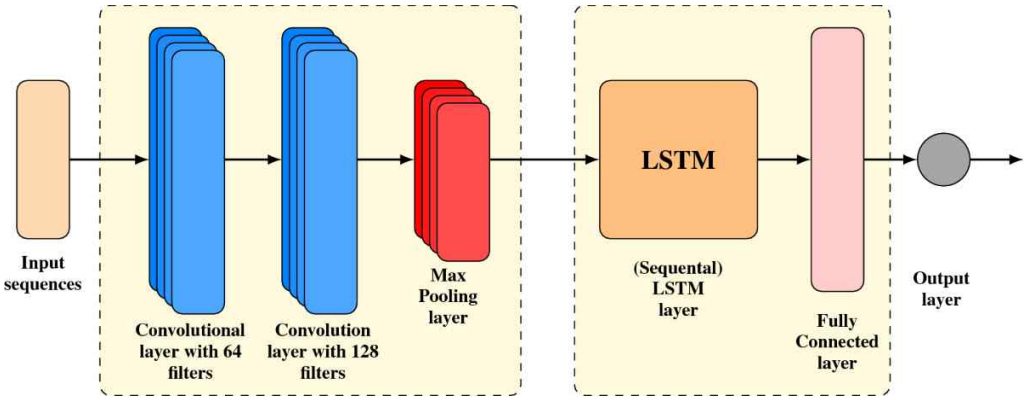
1. 통계적 피처 (평균, 분산, 비율 등)
→ 시간 흐름 반영이 제한적, 동적 패턴 학습 어려움

2. 기존 트리 기반 모델(XGB/LGB)은 **범주형 요인에 강하지만*** [2][3],
시계열 의존성 반영은 어려움

3. 트리의 주요 한계
3-1. 과적합(overfitting): 나무가 지나치게 깊어져 훈련 데이터에만 특화됨.
3-2. 불안정성(instability): 작은 데이터 변화만으로도 구조가 크게 변동할 수 있음

→ 범주형 요인에 강한 특성도 이용하면서 시계열을 보완해줄 모델이 필요

Why CNN-LSTM?



$$y_t = f_{\text{local}}(y_{t-w+1:t}) + g_{\text{long}}(y_{1:t}) + \varepsilon_t$$

CNN-LSTM의 강점

1. CNN (Convolutional Neural Network)
→ 최근 며칠 간 지역적 변동 패턴을 자동 추출* [6]

2. LSTM (Long Short-Term Memory)
→ 장기 의존성과 시간 순서 정보를 학습* [6]

3. 두 모델을 결합하면
→ 급격한 단기 변동 + 장기 주기성을 동시에 포착* [1]

4. 확장 가능성
→ 특정 시점(공휴일·이벤트 등)에 가중치 부여* [7]

→ 단기 변동 + 장기 주기성을 함께 포착하여 더 안정적이고 정밀한 수요 예측 가능

Apply CNN-LSTM

학습 데이터 준비

시계열 윈도우 구성

$$x_t = \log(1 + y_t), \quad X_t = [x_{t-34}, \ldots, x_t] \in \mathbb{R}^{35\times 1}, \quad z_{t+1} = \log(1 + y_{t+1})$$

1. 과거 35일(DL_WIN=35)을 입력 시퀀스로 사용
2. 다음 1일의 매출 로그값을 예측하도록 학습

전처리

$$y_t = \log\big(1 + \max(0, x_t)\big)$$

1. 매출수량 → log1p 변환 (급격한 값의 스케일 안정화)
2. 0 매출일 포함 → 모델이 “판매 없음” 패턴도 학습

→ CNN-LSTM 이 단기+장기 패턴을 윈도우 기반으로 학습할 수 있도록 구조화

Apply CNN-LSTM

모델 아키텍처

CNN (Conv1D, causal padding) → 최근 며칠 간 국소 패턴 추출

$$h_{t,k}^{(c)} = \sigma\Big(\sum_{i=0}^{K-1} w_{k,i}^{(c)} \, x_{t-i} + b_k^{(c)} \Big), \quad t \geq K-1$$

Dropout → 과적합 방지

LSTM (128 hidden units) → 시계열 의존성과 장기 추세 학습

$$\begin{aligned}
i_t &= \sigma\Big(W_i[h_t^{(c)}; h_{t-1}] + b_i\Big), \quad f_t = \sigma\Big(W_f[h_t^{(c)}; h_{t-1}] + b_f\Big), \\
o_t &= \sigma\Big(W_o[h_t^{(c)}; h_{t-1}] + b_o\Big), \quad \tilde{c}_t = \tanh\Big(W_c[h_t^{(c)}; h_{t-1}] + b_c\Big), \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \quad h_t = o_t \odot \tanh(c_t)
\end{aligned}$$

Dense Layer → 최종 예측 (판매량) 산출

$$\hat{z}_{t+1} = W_{\text{out}} h_T + b_{\text{out}}, \quad \hat{y}_{t+1} = \exp(\hat{z}_{t+1}) - 1$$

Apply CNN-LSTM

학습 전략 (MAE·Adam·ES·LR 스케줄)

Loss: MAE (Mean Absolute Error) → 직관적으로 매출 예측 오차를 최소화함

$$\mathcal{L} = \frac{1}{N} \sum_{(X_t, z_{t+1}) \in \mathcal{D}} \left| z_{t+1} - \hat{z}_{t+1} \right|$$

Optimizer: Adam (lr=3e-4) → 안정적인 수렴 달성

$$m \leftarrow \beta_1 m + (1-\beta_1)\nabla_\theta \mathcal{L}, \quad v \leftarrow \beta_2 v + (1-\beta_2)(\nabla_\theta \mathcal{L})^2, \quad \theta \leftarrow \theta - \eta \frac{m/\sqrt{1-\beta_1^t}}{\sqrt{v/(1-\beta_2^t)} + \epsilon}$$

Apply CNN-LSTM

학습 전략 (MAE·Adam·Early Stopping· ReduceLROnPlateau 스케줄)

Callback 활용: EarlyStopping → 과적합 방지

$$\text{If } \min_{e' \le e} \mathcal{L}_{\text{val}}^{(e')} \text{ has no improvement for } p \text{ epochs} \;\Rightarrow\; \text{Stop training.}$$

ReduceLROnPlateau → validation loss 개선 없을 때 학습률 자동 감소

$$\text{if } \mathcal{L}_{\text{val}} \text{ stagnates } (q \text{ epochs}) \;\Rightarrow\; \eta \leftarrow \gamma\,\eta \quad (\gamma < 1)$$

Apply CNN-LSTM

CNN-LSTM 결과 활용

예측 과정: 과거 35일 데이터를 윈도우로 슬라이딩 → 다음날 매출 예측

롤링(roll-out) 방식 → 미래 7일 예측

$$\hat{z}_{t+k+1} = f_\theta(X_t^{(k)}), \quad X_t^{(k)} = [x_{t-34+k}, \ldots, x_t, \hat{z}_{t+1}, \ldots, \hat{z}_{t+k}]$$

Late Fusion

$$\hat{y}_{\text{ens}} = w_A\,\hat{y}_A + (1 - w_A)\,\hat{y}_B, \quad 0 \le w_A \le 1$$

CNN-LSTM 단독 결과 사용 X
XGB/LGB 트리 기반 예측과 가중합 → 더 안정적인 예측값 생성

Apply CNN-LSTM

적용 의의

Tree 기반 모델 → 범주형, 통계적 특성 설명에 강점

CNN-LSTM → 시간 의존성과 급격한 변동 포착에 강점

Late Fusion으로 결합 → 서로 단점 보완 → sMAPE 점수 개선

참고 문헌

[1] "Meenakshi, S., & Ramanathan, P. (2024). Enhanced Retail Sales Forecasting Using Optimized CNN-LSTM Hybrid Model. *Communications on Applied Nonlinear Analysis*, 31(2), 1-12."

[2] Au, T. C. (2018). *Random Forests, Decision Trees, and Categorical Predictors: The "Absent Levels" Problem. Journal of Machine Learning Research*, 19, 1–30.

[3] Lucena, B. (2020). *Exploiting Categorical Structure Using Tree-Based Methods. AISTATS Proceedings*, 108.

[4] Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann.

[5] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. Springer.

[6] Liu, X., & Qi, Y. (2025). Real Time Sales Forecasting in Omnichannel Retail Using a Hadoop Based Hybrid CNN–LSTM Deep Learning Framework. *Journal of Real-Time Data Science*, 8(1), 45-60.

[7] Bhuiyan, M., Rahman, S., Kumar, H., & Lee, D. (2025). Hybrid LSTM-CNN with Attention Mechanism for Accurate and Scalable Grocery Sales Forecasting. *ECCE Proceedings*, 14(3), 210-223.

[8] 이재진,「범주형 변수와 연속형 변수가 혼합된 시계열 데이터 예측 기법」, 인하대학교 석사학위 논문, 2012, 14-22.

개발환경/라이브러리 버전

개발환경

- OS: Windows 11 (10.0.26100)
- Python: 3.13.6 (64-bit, VS Code)

핵심 라이브러리 버전

- numpy 2.3.2
- pandas 2.3.1
- scikit-learn 1.7.1
- xgboost 3.0.4
- lightgbm 4.6.0
- tensorflow 2.20.0 (CPU)
- torch 2.8.0+cpu