식음업장 메뉴 수요 예측 AI 온라인 해커톤

- 링크: https://dacon.io/competitions/official/236559/overview/description
- 유형: LG Aimers | 채용 | 알고리즘 | 정형 | 시계열 | 수요 예측 | SMAPE

1. 개요

A. 배경

> 리조트 내 식음업장은 계절, 요일, 투숙객 수, 행사 일정 등 다양한 요인에 따라 수요가 크게 변동하는 환경에 놓여 있습니다. 특히 휴양지 리조트는 단기간에 집중되는 고객 수요와 예측하기 어려운 방문 패턴으로 인해, 메뉴별 식자재 준비, 인력 배치, 재고 관리에 있어 높은 운영 난이도를 가집니다.
>
> 이러한 복잡한 운영 환경 속에서 정확한 메뉴 수요 예측은 비용 절감과 고객 만족도 향상에 있어 핵심적인 요소로 작용합니다. 최근에는 AI 기술을 활용한 수요 예측이 식음 서비스 운영의 새로운 해법으로 주목받고 있으며, 정형화된 과거 매출 데이터와 외부 요인을 함께 분석하는 방식이 빠르게 확산되고 있습니다.
>
> 이번 해커톤은 리조트 내 식음업장에서의 실전 수요 예측 문제를 AI로 해결해보는 것을 목표로 합니다. Aimers 여러분들은 실제 식음업장에서 수집된 판매 데이터를 기반으로, 각 메뉴가 1주일 동안 얼마나 판매될지를 예측하는 모델을 개발하게 됩니다. 이를 통해 데이터 기반 의사결정이 리조트 운영에 어떤 가치를 더할 수 있는지를 직접 체감할 수 있을 것입니다.

B. 주제: 리조트 내 식음업장 메뉴별 1주일 수요 예측 AI 모델 개발 - 리조트 식음업장에서 수집된 과거의 메뉴별 판매 데이터를 기반으로, 향후 1주일간 각 메뉴의 예상 판매량을 예측하는 AI 모델을 개발

C. 규칙

i. 평가산식: 식음업장 별 가중치가 있는 SMAPE

- s: 식음업장명
- w_s: 식음업장 s의 가중치 (비공개)
- I_s: 식음업장 s에 속한 품목 컬럼 집합
- T_i: 품목 i에서 유효한 날짜 수 ($A_{t,i} \neq 0$)
- $A_{t,i}, P_{t,i}$: 날짜 t, 품목 i의 실제값과 예측값

1. 각 식음업장 별 가중치가 존재하며, '담하'와 '미라시아'는 다른 업장보다 높은 가중치로 반영됩니다. (단, 업장 별 가중치 값은 공개하지 않습니다.)
2. 실제 매출 수량이 0인 경우에는 평가 산식 계산에 반영되지 않습니다.
3. Public score : 전체 테스트 데이터 샘플 중 사전 샘플링된 50%
4. Private score : 전체 테스트 데이터 샘플 100%

ii. 외부 데이터 및 사전 학습 모델 관련 규칙

1. 사전학습모델 사용 가능 범위
   공식적으로 가중치가 공개되었으며, 최소한 비상업적 이용이 허용된 오픈소스 라이선스(MIT, Apache 2.0 등) 하에 배포된 모델만 사용 가능합니다.
   해당 조건을 만족하지 않는 모델 및 가중치는 사용할 수 없습니다.
2. API 사용 제한
   원격 서버 기반의 API 형태로만 접근 가능한 모델(OpenAI API, Gemini API 등)은 사용이 불가합니다.
   모든 모델은 로컬 환경에서 직접 실행 가능해야 하며, 외부 서버에 의존하는 방식은 제한됩니다.
3. 외부 데이터 사용 금지
   경진대회에서 제공하는 공식 데이터 외의 외부 데이터는 사용할 수 없습니다.

iii. 시계열 예측 관련 Data Leakage 방지 규칙

1. 평가 데이터는 학습에 사용할 수 없습니다.
   평가 Input(28일), Target(7일)은 어떤 경우에도 모델 학습에 활용할 수 없습니다.
   Pseudo Labeling 등 추론 결과를 이용한 재학습도 불가합니다.
   평가 데이터는 오직 추론 시점에서 입력으로만 사용할 수 있습니다.
2. 추론 시 평가 Input으로 제공된 28일 외의 데이터를 추가로 사용할 수 없습니다.
   각 평가 샘플에는 Input으로 28일간의 시계열 데이터만 제공되며, 예측 시에는 해당 구간만을 모델 입력으로 사용해야 합니다.
   평가 시점에서 Lookback 기간을 임의로 확장하거나, 추가적인 과거 데이터를 연결하여 사용하는 것은 허용되지 않습니다.
   즉, 모든 평가 샘플은 제공된 28일 데이터를 기준으로만 예측이 이루어져야 하며, 모델 구조나 데이터 처리 방식에 관계없이 28일을 초과한 입력 사용은 금지됩니다.
3. 평가 샘플은 서로 독립적으로 추론해야 합니다.
   하나의 평가 샘플 결과나 입력을 다른 샘플의 예측에 사용하는 것은 금지됩니다.
   모든 샘플은 각자의 Input(28일)만을 사용해 개별적으로 예측되어야 합니다.
4. 추론 시점 이후 정보는 사용할 수 없습니다.
   각 평가 샘플의 추론 시점은 Input의 마지막 날짜입니다.
   이 시점을 기준으로 이후의 데이터(예측 대상 포함)는 모두 미래로 간주되며 활용할 수 없습니다.
5. 외부 데이터는 사용할 수 없지만, 도메인 지식은 활용할 수 있습니다.
   대회에서 제공한 데이터 외의 외부 데이터, 크롤링, API 호출 등은 금지됩니다.
   다만, 도메인 지식 기반의 정보(예: 공휴일, 요일 등)는 활용 가능합니다. 다만, 도메인 지식 기반의 정보를 습득할 수 있는 시점도 추론 시점에 유의하여 활용할 수 있어야 합니다.
   예: "5월 5일은 공휴일이다", "일요일은 주말이다" → 추론 시점 이전에 알 수 있는 도메인 지식 기반의 정보이므로 사용 가능

2. 데이터 구성

A. train [폴더]

```
영업일자,영업장명_메뉴명,매출수량
2023-01-01,느티나무 셀프BBQ_1인 수저세트,0
2023-01-02,느티나무 셀프BBQ_1인 수저세트,0
2023-01-03,느티나무 셀프BBQ_1인 수저세트,0
2023-01-04,느티나무 셀프BBQ_1인 수저세트,0
```

i. train.csv [파일]

ii. 2023.01.01 ~ 2024.06.15의 영업장명_메뉴명별 매출 수량 정보

iii. 영업일자

iv. 영업장명_메뉴명

v. 매출수량

B. test [폴더]

```
영업일자,영업장명_메뉴명,매출수량
2025-04-27,느티나무 셀프BBQ_1인 수저세트,0
2025-04-28,느티나무 셀프BBQ_1인 수저세트,0
2025-04-29,느티나무 셀프BBQ_1인 수저세트,2
2025-04-30,느티나무 셀프BBQ_1인 수저세트,0
```

i. TEST_00.csv ~ TEST_09.csv [파일]

ii. 2025년의 특정 시점(28일)의 영업장명_메뉴명별 매출 수량 정보

iii. 영업일자

iv. 영업장명_메뉴명

v. 매출수량

C. sample_submission.csv [파일] - 제출 양식

```
영업일자,느티나무 셀프BBQ_1인 수저세트,느티나무 셀프BBQ_BBQ55(단체),"느티나무 셀프BBQ_대여료 30,000원","느티나무 셀프BBQ_대여료 60,000원"
TEST_00+1일,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
TEST_00+2일,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
TEST_00+3일,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

i. 각 영업장명_메뉴명의 TEST 파일별 +1일, +2일,..., +7일의 매출수량 예측 결과

ii. 영업일자 : TEST_00+1일, TEST_00+2일, TEST_00+3일 ... TEST_09+1일, TEST_09+2일, TEST_09+7일

3. 사전 연구
   A. 주요 방법론/프레임워크 정리
      i. eXtreme Gradient Boosting
         1. 기존 Gradient Tree Boosting 알고리즘에 과적합 방지를 위한 기법이 추가된 학습 알고리즘
            A. Gradient Tree Boosting: 라벨 또는 종속 변수가 있는 데이터를 학습하고 이를 통해 새로운 데이터의 라벨을 예측하는 지도 학습 알고리즘
         2. 기본 학습기를 Decision Tree로 하여 Gradient Boosting과 같이 Gradient(잔차)를 이용해 이전 모형의 약점을 보완하는 방식으로 학습
            A. Decision Tree: 의사 결정 규칙과 그 결과들을 트리 구조로 도식화한 도구
            B. Gradient: 관찰된 값과 관심 수량의 추정된 값 간의 차이
         3. 장점
            A. 높은 성능: 대부분의 테이블 데이터에서 우수한 성능
            B. 빠른 훈련: 병렬 처리 및 최적화된 구현
            C. 과적합 방지: 다양한 정규화 기법 내장
            D. 결측치 처리: 자동으로 결측치 처리
            E. 특성 중요도: 해석 가능한 특성 중요도 제공
         4. 단점
            A. 파라미터 복잡: 많은 하이퍼파라미터 튜닝 필요
            B. 메모리 사용량: 큰 데이터셋에서 메모리 집약적
            C. 선형 관계: 단순한 선형 관계에서는 과도할 수 있음
         5. 예제

```python
import pandas as pd
import numpy as np
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error

# 데이터 준비 (당신의 시계열 데이터 예시)
# 특성 엔지니어링이 완료된 상태라고 가정
X = daily_total_sales[['sales_lag_1', 'sales_lag_7', 'ma_7', 'ma_30', 'weekday']]
y = daily_total_sales['판매량']

# 시계열 특성상 순서 유지하며 분할
split_idx = int(len(X) * 0.8)
X_train, X_test = X[:split_idx], X[split_idx:]
y_train, y_test = y[:split_idx], y[split_idx:]

# 모델 훈련
xgb = XGBRegressor(
    n_estimators=1000,
    learning_rate=0.01,
    max_depth=6,
    min_child_weight=1,
    subsample=0.8,
    colsample_bytree=0.8,
    reg_alpha=0.1,
    reg_lambda=1.0,
    random_state=42,
    n_jobs=-1  # 멀티코어 사용
)

xgb.fit(X_train, y_train)

# 예측 및 평가
y_pred = xgb.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
```

6. 추가 팁
   A. 시간 순서 보장: 교차검증시 반드시 TimeSeriesSplit 사용
   B. 특성 엔지니어링 중요: 지연 특성, 이동평균, 계절성 특성 등 활용
   C. 조기 종료 활용: 과적합 방지를 위해 early_stopping_rounds 사용
   D. 정규화 파라미터 조정: reg_alpha, reg_lambda로 과적합 제어
   E. 앙상블과 결합: 다른 모델들과 앙상블로 성능 향상

ii. Temporal Embedding

1. 학습을 통해 의미적으로 유사한 시간들을 가까운 벡터로 매핑하는 기법
2. 벡터 효율성: 원-핫 인코딩과 비교해서 모든 시간 정보를 정해진 차원으로 압축하면서도 더 풍부한 표현이 가능
   A. 원-핫 인코딩: 범주형 데이터를 기계가 이해할 수 있는 수치형 벡터로 변환하는 기법
3. 예시
   A. 월요일과 화요일은 비슷한 패턴 (주중)
   B. 토요일과 일요일은 비슷한 패턴 (주말)
   C. 12월과 1월은 비슷한 패턴 (연말연시)
4. 예제

```python
class AdvancedSalesPredictor(nn.Module):
    def __init__(self, d_model=128, lstm_hidden=64):
        super().__init__()

        # 시간 임베딩
        self.temporal_embed = TemporalEmbedding(d_model)

        # 판매량 자체도 임베딩 (선택적)
        self.sales_proj = nn.Linear(1, d_model)

        # LSTM
        self.lstm = nn.LSTM(d_model * 2, lstm_hidden,
                            num_layers=2, batch_first=True, dropout=0.2)

        # 출력 레이어
        self.fc = nn.Sequential(
            nn.Linear(lstm_hidden, 32),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(32, 1)
        )

    def forward(self, sales_data, time_data):
        batch_size, seq_len = sales_data.shape[:2]

        # 시간 임베딩
        temporal_emb = self.temporal_embed(time_data)  # [batch, seq, d_model]

        # 판매량 임베딩
        sales_emb = self.sales_proj(sales_data)  # [batch, seq, d_model]

        # 결합
        combined = torch.cat([sales_emb, temporal_emb], dim=-1)  # [batch, seq, d_model*2

        # LSTM
        lstm_out, _ = self.lstm(combined)

        # 예측
        prediction = self.fc(lstm_out[:, -1, :])

        return prediction

# 모델 사용
model = AdvancedSalesPredictor(d_model=64)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

iii. Stacked LSTM

1. 여러 개의 LSTM 레이어를 위로 쌓아 시퀀스 표현을 점점 더 높은 수준으로 추상화하도록 만든 구조
2. 구조
   A. 초기화
      i. 1층: 원시 패턴 학습 (단기 의존성)
      ii. 2층: 중간 수준 패턴 (주간, 월간 패턴)
      iii. 3층: 고수준 패턴 (계절성, 장기 트렌드)
   B. 전진
      i. 1층: 기본 시계열 패턴
      ii. 2층: 복합 패턴
      iii. 3층: 고수준 추상화
3. 장점: 표현 능력이 커져 복잡한 시퀀스 패턴을 더 잘 모델링함
4. 단점: 계산/메모리 비용 증가, 과적합/학습 불안정성 가능, 너무 깊으면 이득이 적음
5. 예제

```python
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error

class StackedLSTM(nn.Module):
    def __init__(self, input_size, hidden_sizes, output_size=1, dropout=0.2):
        super(StackedLSTM, self).__init__()

        self.hidden_sizes = hidden_sizes
        self.num_layers = len(hidden_sizes)

        # LSTM 레이어들을 ModuleList로 관리
        self.lstm_layers = nn.ModuleList()

        # 첫 번째 LSTM 레이어
        self.lstm_layers.append(nn.LSTM(input_size, hidden_sizes[0],
                                        batch_first=True, dropout=dropout))

        # 나머지 LSTM 레이어들
        for i in range(1, len(hidden_sizes)):
            self.lstm_layers.append(nn.LSTM(hidden_sizes[i-1], hidden_sizes[i],
                                            batch_first=True, dropout=dropout))

        # Dropout 레이어들
        self.dropout = nn.Dropout(dropout)

        # 출력 레이어
        self.fc = nn.Linear(hidden_sizes[-1], output_size)

    def forward(self, x):
        # x shape: (batch_size, seq_len, input_size)

        # 각 LSTM 레이어를 순차적으로 통과
        for i, lstm_layer in enumerate(self.lstm_layers):
            x, _ = lstm_layer(x)
            x = self.dropout(x)

        # 마지막 시점의 출력만 사용
        output = self.fc(x[:, -1, :])  # (batch_size, output_size)

        return output

# 모델 생성 예시
model = StackedLSTM(
    input_size=1,          # 판매량 하나의 특성
    hidden_sizes=[64, 32, 16],  # 3층 LSTM (64 → 32 → 16)
    output_size=1,         # 판매량 예측
    dropout=0.3
)

print(model)
```

iv. Light GBM

1. 다른 GBM 알고리즘(XGBoost 등)과 다르게 level-wise 트리(수평 트리) 확장 구조 대신 leaf-wise 트리(수직 트리) 확장 구조를 사용하여 속도와 메모리를 향상
2. 특성
   A. 범주형 변수가 숫자형 변수로 변환되지 않고 그대로 학습됨
   B. GPU 활용 가능
   C. 일부 파라미터가 XGBoost와 호환 가능
3. 최적의 파라미터 탐색: Bayesian Optimization
4. 예제

```python
import pandas as pd
import numpy as np
from lightgbm import LGBMRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error

# 데이터 준비 (당신의 시계열 데이터 예시)
# 특성 엔지니어링이 완료된 상태라고 가정
X = daily_total_sales[['sales_lag_1', 'sales_lag_7', 'ma_7', 'ma_30', 'weekday']]
y = daily_total_sales['판매량']

# 시계열 특성상 순서 유지하며 분할
split_idx = int(len(X) * 0.8)
X_train, X_test = X[:split_idx], X[split_idx:]
y_train, y_test = y[:split_idx], y[split_idx:]

# 모델 훈련
lgbm = LGBMRegressor(
    n_estimators=1000,        # 트리 개수
    learning_rate=0.01,       # 학습률
    max_depth=6,              # 최대 트리 깊이
    min_child_samples=1,      # 리프 노드가 가져야 할 최소한의 샘플 가중치 합
    subsample=0.8,            # 데이터 샘플링 비율 (row sampling)
    feature_fraction=0.8,     # 각 트리별 컬럼 샘플링 비율
    reg_alpha=0.1,            # L1 정규화
    reg_lambda=1.0,           # L2 정규화
    random_state=42,          # 랜덤 시드
    n_jobs=-1                 # 멀티코어 사용
)

lgbm.fit(X_train, y_train)

# 예측 및 평가
y_pred = lgbm.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
```

5. 장점
   A. 적은 시간: 학습에 적은 시간이 소요(XGBoost 학습속도의 1.3~1.5배)
   B. 메모리 사용량: 상대적으로 적음
   C. 분할 능력: 카테고리형 feature의 자동 변환과 최적 분할
   D. 처리 능력: 대용량 데이터 처리 가능
6. 단점
   A. 오버피팅: 적은 데이터셋(10000건 이하)에서는 오버피팅 가능

v. Optuna: 하이퍼파라미터 최적화 라이브러리

```python
"max_depth": trial.suggest_int("max_depth", 4, 12), # 트리 최대 깊이
"learning_rate": trial.suggest_float("learning_rate", 0.01, 0.3, log=True), # 학습률
"subsample": trial.suggest_float("subsample", 0.6, 1.0), # 데이터 샘플 비율
"colsample_bytree": trial.suggest_float("colsample_bytree", 0.6, 1.0), # 특성 샘플 비율
"gamma": trial.suggest_float("gamma", 1e-8, 1.0, log=True), # 최소 손실 감소 값
"lambda": trial.suggest_float("lambda", 1e-8, 1.0, log=True), # L2 정규화항
"alpha": trial.suggest_float("alpha", 1e-8, 1.0, log=True), # L1 정규화항
```

B. 분류 기초 방안 구상

- 분류 목적: 메뉴별로 실제 주요 소비자층이 언제/어떻게 이 메뉴를 소비하는가를 반영
- 분류 기준
  1. 주중평균: 평일(월~금) 기준 평균 매출수량
  2. 주말평균: 주말(토/일) 기준 평균 매출수량
  3. WR (Weekend Ratio): 주말평균 / 주중평균
  4. Lv(Level): High(주중평균 ≥ 10), Mid(1 ≤ 주중평균 < 10), Low(주중평균 < 1)로 구분
- 분류별 의미

세그먼트	의미(주요 소비자)	분류 기준
Family/Tourist	주말에 가족/관광객 수요 급증 메뉴	WR ≥ 1.x, Lv = Mid / High
Corporate/Group	주중에 기업/단체 예약 수요 중심 메뉴	WR ≤ 0.x, Lv = High
Premium/Repeat	평일·주말 모두 대량 고정수요/프리미엄	0.x < WR < 1.x, Lv = High
Everyday/Office	일상/직장인 등 꾸준한 중간 수요	0.x < WR < 1.3, Lv = Mid
Unclassified	위에 해당 안 됨	위 조건 제외

- 파생 변수

피처 종류	변수명	설명
날짜 정보	요일, 주말, 월, 분기, 연도	요일별/월별/계절별 패턴 포착
이동평균·이동표준편차	mean_lag7, std_lag30	최근 7일/30일 평균·분산
시계열 라그(직전 수량)	lag_1, lag_7, lag_30	전날/전주/전월의 수량
누적합·전년동월	cumsum, prev_year	트렌드, 계절성 반영
세그먼트	Segment	소비자 유형 (앞서 분류한 것 활용)

C. 개선 방안 구상

i. Feature 엔지니어링

1. 날짜/캘린더 기반 파생
   - A. 기본
     - i. 계절, 월, 분기, 연도, 요일, 주말/평일 플래그
     - ii. 공휴일, 연휴, 영업일/비영업일, 성수기/비성수기
     - iii. 월초/말, 분기초/말, 연속 휴일 여부 (ex: 3일 연휴)
   - B. 과거 비교
     - i. 지난달/작년 동월 매출과의 비율 및 증감률
   - C. 특정일/이벤트

i. 명절, 프로모션, 코로나 등 주요 이벤트 flag

D. 월별/계절별 판매 패턴

i. 메뉴별/영업장별 월별, 계절별 매출 피처

ii. 월/계절별 잘 팔리는 메뉴, 시즌성 분석

2. 트랜드/패턴 파생

A. 최근 k일(혹은 7, 30, 90일 등)의 이동평균/이동표준편차, 최대·최소, 변동계수(CV) 등

B. 단순 이동평균뿐 아니라 기하/지수적 이동평균 등 다양한 smoothing 기법

C. 전일/전주/전월 대비 증감(Percent Change)

D. 누적 성장률 (누적합, 누적평균, 누적 매출 증가율 등)

E. 이벤트/프로모션에 의한 변동 마킹(플래그)

F. 판매량 0인 날 비율: 매출 0일 비율 높은 메뉴 필터링(예측모델에서 제외 또는 flag)

3. 수요/매출 패턴 특성

A. 상위 TOP 메뉴/주력 메뉴: 영업장별/전체 메뉴별 판매량 상위 N개 메뉴 플래그, 순위 등

B. 비인기 메뉴 마킹: 매출 0일 비율, 판매 저조 메뉴 필터링

C. 일별 총 방문객수 및 1인당 평균 판매수: 실데이터 있으면 merge, 없으면 메뉴 판매량 총합/가정값(1.5 등)으로 추정

D. 품절/판매중지/신제품/단종 등 특수상황 변수

4. 메뉴/영업장 특성

A. 메뉴 유형/카테고리 (식사/디저트/음료/주류 등, TF-IDF, KMeans 등 활용)

i. 메뉴별 Type 신규 feature

ii. 시간대별 유형별 매출패턴 (ex: 아침-커피, 저녁-맥주 등)

iii. 메뉴별 가격대 구간화

B. 영업장 특성: 위치(내부/외부/야외 등), 규모, 유형(푸드코트/레스토랑/카페 등)

C. TOP/비TOP 메뉴, 시즌성, 시간대/요일별 인기 등 추가 flag

5. 기타 특성

A. 성수기/비성수기 플래그 (월별, 이벤트별, 방학/휴가철 등)

B. 1인당 메뉴 구매량 추정 (방문객수와 연계)

C. 계절별/월별 특화 메뉴 flag

D. 특수상황/이벤트 마킹 (공급 차질, 대외 변수 등)

ii. 범주형/수치형 명확 분리

1. 범주형(Categorical)

A. 영업장명_메뉴명

B. 메뉴 카테고리

C. 요일, 월, 분기

D. 세그먼트

E. 이벤트/공휴일 플래그
F. 가격대 구간 등

2. 수치형(Numeric)
A. 매출수량, lag, 이동평균, 표준편차
B. 변동률, 누적합, 이동변동계수, 최근 n일/주간 증감
C. 가격, 성장률 등

D. 수요 패턴 분석

i. ADI (Average Demand Interval)
1. 수요가 발생하는 간격의 평균을 의미
2. 계산식: ADI = 총 관찰 기간 / 수요 발생 횟수
3. 값이 클수록 수요가 간헐적(intermittent)임을 의미

ii. CV (Coefficient of Variation)
1. 수요의 변동성을 나타내는 지표
2. 계산식: CV = 표준편차 / 평균 (수요가 있는 기간만 고려)
3. 값이 클수록 수요의 변동성이 큼

iii. 수요 패턴 분류 (Syntetos & Boylan, 2005)
1. Smooth (ADI ≤ 1.32, CV^2 ≤ 0.49): 안정적인 수요
2. Intermittent (ADI > 1.32, CV^2 ≤ 0.49): 간헐적이지만 일정한 수요
3. Erratic (ADI ≤ 1.32, CV^2 > 0.49): 규칙적이지만 변동성이 큰 수요
4. Lumpy (ADI > 1.32, CV^2 > 0.49): 간헐적이고 변동성이 큰 수요

iv. 분석 결과
1. Lumpy (54.4%): 가장 복잡한 패턴, ML 기법 필요
2. Intermittent (31.6%): Croston's Method 적용 가능
3. Erratic (13.0%): Robust 예측 모델 필요
4. Smooth (1.0%): 전통적 시계열 모델 적용

E. 메뉴 분류

i. 포레스트릿: 12 | 305421
1. 음식 꼬치어묵, 떡볶이, 치즈 핫도그, 페스츄리 소시지
2. 음료(공용) 생수, 스프라이트, 코카콜라
3. 음료(하계) 복숭아 아이스티, 아메리카노(ICE), 카페라떼(ICE)
4. 음료(동계) 아메리카노(HOT), 카페라떼(HOT)

ii. 카페테리아: 24 | 240850
1. 음식(단일) 돼지고기 김치찌개, 새우 볶음밥, 새우튀김 우동, 등심 돈까스, 약 고추장 돌솥비빔밥, 어린이 돈까스, 진사골 설렁탕, 짜장면, 짜장밥, 짬뽕, 짬뽕밥, 치즈 돈까스
2. 음식(옵션) 공깃밥(추가)
3. 음식(단체) 한상 삼겹구이 정식(2인), 단체식 13000(신), 단체식 18000(신)
4. 음료(하계) 아메리카노(ICE), 카페라떼(ICE), 복숭아 아이스티, 구슬아이스크림

5. 음료(동계) 수제 아메리카노(HOT), 카페라떼(HOT)
6. 기타 오픈푸드, 샷 추가, 소요시간 약 15~20분

iii. 화담숲주막: 8 | 146311
1. 음식 병천순대, 해물파전
2. 음료 스프라이트, 찹쌀식혜, 콜라
3. 주류 느린마을 막걸리, 단호박 식혜, 참살이 막걸리

iv. 담하: 42 | 124917
1. 음식(단일) (정식) 된장찌개, (정식) 물냉면, (정식) 비빔냉면, (후식) 된장찌개, (후식) 물냉면, (후식) 비빔냉면, 갑오징어 비빔밥, 갱시기, 꼬막 비빔밥, 담하 한우 불고기, 담하 한우 불고기 정식, 더덕 한우 지짐, 들깨 양지탕, 봉평메밀 물냉면, 생목살 김치찌개, 은이버섯 갈비탕, 한우 떡갈비 정식, 한우 미역국 정식, 한우 우거지 국밥, 한우 차돌박이 된장찌개, 황태해장국, 명태회 비빔냉면
2. 음식(단체) (단체) 공깃밥, (단체) 생목살 김치전골 2.0, (단체) 은이버섯 갈비탕, (단체) 한우 우거지 국밥, (단체) 황태해장국 3/27까지
3. 음식(옵션) 공깃밥, 라면사리, 메밀면 사리
4. 음료 스프라이트, 제로콜라, 콜라
5. 주류 느린마을 막걸리, 명인안동소주, 문막 복분자 칵테일, 참이슬, 처음처럼, 카스, 테라, 하동 매실 칵테일
6. 기타 룸 이용료

v. 미라시아: 31 | 98071
1. 음식(단일) (오븐) 하와이안 쉬림프 피자, (화덕) 불고기 페퍼로니 반반피자, BBQ Platter, 브런치(대인) 주말, 브런치(대인) 주중, 브런치(어린이), 쉬림프 투움바 파스타, 보일링 랍스타 플래터, 보일링 랍스타 플래터(덜매운맛), 오븐구이 윙과 킬바사소세지, 칠리 치즈 프라이, 콥 샐러드
2. 음식(단체) 미라시아 브런치 (패키지), 브런치 2인 패키지, 브런치 4인 패키지, (단체)브런치주중 36,000
3. 음식(옵션) BBQ 고기추가, 공깃밥, 파스타면 추가(150g)
4. 음료 스프라이트, 애플망고 에이드, 코카콜라, 코카콜라(제로), 핑크레몬에이드
5. 주류 글라스와인 (레드), 레인보우칵테일(알코올), 버드와이저(무제한), 스텔라(무제한), 얼그레이 하이볼, 유자 하이볼, 잭 애플 토닉

vi. 느티나무 셀프BBQ: 23 | 69786
1. 음식(단체) BBQ55(단체), 본삼겹 (단품,실내)
2. 음식(옵션) 신라면, 쌈야채세트, 쌈장, 육개장 사발면, 햇반, 허브솔트
3. 음료 콜라 (단체), 스프라이트 (단체)
4. 주류 참이슬 (단체), 카스 병(단체)
5. 장소 대여료 30,000원, 대여료 60,000원, 대여료 90,000원, 잔디그늘집 대여료 (12인석), 잔디그늘집 대여료 (6인석)
6. 소품 1인 수저세트, 일회용 소주컵, 일회용 종이컵, 잔디그늘집 의자 추가, 친환경

접시 14cm, 친환경 접시 23cm

vii. 화담숲카페: 5 | 62634

1. 음료(공용) 메밀미숫가루
2. 음료(하계) 아메리카노 ICE, 카페라떼 ICE, 현미뻥스크림
3. 음료(동계) 아메리카노 HOT

viii. 연회장: 23 | 28373

1. 음식(단일) 골뱅이무침, 돈목살 김치찌개 (밥포함), 로제 치즈떡볶이, 마라샹궈, 매콤 무뼈닭발&계란찜, 왕갈비치킨, Cookie Platter
2. 음식(단체) 모둠 돈육구이(3인)
3. 음식(옵션) 삼겹살추가 (200g), 야채추가, 공깃밥, 주먹밥 (2ea)
4. 음료 Regular Coffee
5. 주류 Cass Beer
6. 장소 Conference L1, Conference L2, Conference L3, Conference M1, Conference M8, Conference M9, Convention Hall, Grand Ballroom, OPUS 2

ix. 라그로타: 25 | 17453

1. 음식(단일) 그릴드 비프 샐러드, 까르보나라, 모둠 해산물 플래터, 미션 서드 카베르네 쉬라, 버섯 크림 리조또, 시저 샐러드, 알리오 에 올리오, 양갈비 (4ps), 해산물 토마토 리조또, 해산물 토마토 스튜 파스타, 해산물 토마토 스파게티
2. 음식(옵션) AUS (200g), 빵 추가 (1인), 한우 (200g)
3. 음료 스프라이트, 아메리카노, 자몽리치에이드, 제로콜라, 콜라
4. 주류 Gls.Sileni, Gls.미션 서드, 카스, 하이네켄(생)
5. 기타 Open Food, G-Charge(3)

4. 일자별 대표 개발 및 결과
   A. 0811_LGBM: 0.5998
      i. 팀원이 구성한 기본 LGBM 코드를 베이스로 시작
      ii. 공휴일 목록을 추가하여 피처로 입력
      iii. 테스트용 공휴일 SMAPE 함수 추가
      iv. 특성 추가로 인한 성능 저하를 확인
   B. 0812_LGBM: 0.5902
      i. Optuna 라이브러리를 적용해 파라미터 최적화 시도
      ii. 한 번 돌리는데 시간이 매우 오래 걸리기 때문에(약 3시간) 최적화 필요
   C. 0813_XGBoost: 0.5691
      i. Github 코드를 기반으로 XGBoost 기법 도입
      ii. 메뉴 별 이상치 값(상식적인 범위 밖에 있는 값)들을 IQR(사분범위) 계산을 통해 최대/최소값으로 대체
      iii. 날짜 별 주기성 각도, 래그 및 롤링 특징 생성
      iv. 예측할 날짜의 프레임 별로 래그 및 롤링 특징 계산하여 예측 수행
   D. 0815_XGBoost: 0.5324
      i. 중요도가 강조된 일부 키워드에 강제적인 가중치를 부여
      ii. '담하'와 '마라시아'에 2배의 가중치 부여
   E. 0817_XGBoost: 0.5668
      i. 가중치 부여 코드 기반으로 일부 피처 추가
      ii. 피처 추가로 인한 성능 저하를 확인
   F. 0819_XGBoost: 0.5220
      i. 2024년 데이터에 가중치 추가
      ii. 매출량이 0인 데이터에 가중치 차감
   G. 0822_XGBoost
      i. 주요 가중치 파라미터 조절
   H. 0824_XGBoost
      i. 학습 데이터 기간 내에서 무작위로 시작점을 선택하고 정해진 기간만큼을 학습하는 윈도우를 설정
      ii. 학습 데이터 윈도우 바로 다음 7일을 검증 데이터로 사용
      iii. 이 과정을 N번 반복하여 N개의 모델을 학습시키고 리스트에 저장해 최종 예측 시 N개의 모든 모델로 예측을 수행한 후 그 결과의 평균을 최종 예측값으로 사용

5. 결론
   A. 피처 엔지니어링의 효과
      i. 이상치를 IQR 값으로 대체하고 날짜 관련 파생 변수(주기성, 래그, 롤링)를 생성한 모델에서 성능이 크게 향상되는 것을 확인
      ii. 모델 자체의 성능뿐만 아니라 데이터의 특성을 잘 반영하는 피처를 생성하는 것이 예측 정확도에 매우 중요
   B. 가중치 부여의 역할
      i. 가장 뛰어난 성능을 보인 모델은 특정 키워드, 최신 데이터(2024년), 매출이 0인 데이터에 대한 가중치를 부여한 결과
      ii. 단순히 피처를 추가하는 것보다 데이터의 중요도에 따라 가중치를 조절하는 것이 성능 향상에 더 효과적
   C. 앙상블 기법
      i. 최종적으로 시도된 윈도우 기반의 앙상블 학습은 최고 기록에 근접하였음
      ii. 이를 토대로 모델을 더 정제할 시 높은 예측을 얻을 수 있을 것으로 기대됨