

[KDT] 기업 맞춤형 AI 융복합 인재 양성과정

편의점 수요예측 기반

발주/물류 최적화

머신러닝 프로젝트

TEAM 사공사오

김요원, 김금, 정연덕, 최성민

INDEX

01

프로젝트 개요

- 추진 배경
- 개발 목표
- 팀 구성 및 역할
- 개발 환경

02

데이터 분석

- 데이터 수집
- 데이터 탐색적 분석
(EDA)

03

예측 모델링

- 모델 선택
- Linear Regression
- Random Forest
- XGBoost
- LightGBM
- 모델 평가

04

웹서비스 구현

- 구현 방향
- Streamlit 코드
- UI 구현

05

결론 및 평가

- 기대 효과
- 자체 피드백

1. 추진 배경

"데이터 기반 수요 예측으로 편의점 발주와 재고를 최적화하는 머신러닝 프로젝트"

재고 과잉 (폐기)

유통기한 경과로 인한 비용 손실 발생

재고 부족 (품절)

판매 기회 상실 및 고객 만족도 하락



머신러닝 기반 해결 방향 - 데이터 기반 수요 예측



판매 데이터를 학습하여 정확한 발주량 산출 □ 재고 과잉과 품절을 동시에 최소화하는 전략적 발주 지원



수요량 예측 데이터를 활용하여 쉽게 발주 결정 가능 □ 운영 효율성 극대화

1. 개발 목표



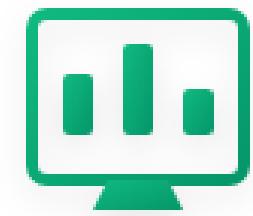
AI 수요 예측

머신러닝을 활용한
품목별/일별 판매량 정밀 예측



자동 발주 추천

예측 데이터 기반
최적 발주 수량 자동 제안



통합 대시보드

매출, 재고, 예측 정보를
한눈에 파악하는 시각화 웹

1. 팀 구성 및 역할



김요원

※ 팀장

- 데이터 기획
- 모델학습(Linear Regression, XGBoost)
- UI 구현, 배포
- 문서작업(전체)



김금

※ 팀원

- 데이터 설계 및 수집
- 모델학습(Random Forest)
- 배포
- 문서작업(데이터 분석)



최성민

※ 팀원

- 데이터 전처리
- 모델학습(LightGBM)
- 배포
- README 파일 작업



정연덕

※ 팀원

- 데이터 전처리
- 모델학습(Random Forest)
- 배포
- 문서작업(모델 학습)

1. 개발 환경

Programming & Environment



python



scikit-learn

Machine Learning Models



NumPy



LightGBM

Model Storage



Hugging Face



Figma

2. 데이터 수집

데이터 확보 전략

기업 보안상 실제 매출 데이터 접근이 어려워,

ChatGPT 시뮬레이션 데이터를 (5개의 지역 각

10개 총 50개의 매장의 10개의 물품의 1년간의

수요 데이터) 활용했습니다.

날짜	요일	주말여부	지역	매장	물품	온도	날씨	수요물품수
2025-01-01	수		0 서울	서울 강남역점	아이스 음료	17.8 맑음		308
2025-01-01	수		0 서울	서울 강남역점	생수	17.8 맑음		161
2025-01-01	수		0 서울	서울 강남역점	주류	17.8 맑음		246
2025-01-01	수		0 서울	서울 강남역점	마스크	17.8 맑음		134
2025-01-01	수		0 서울	서울 강남역점	컵라면	17.8 맑음		190
2025-01-01	수		0 서울	서울 강남역점	도시락	17.8 맑음		176
2025-01-01	수		0 서울	서울 강남역점	핫음료	17.8 맑음		222
2025-01-01	수		0 서울	서울 강남역점	아이스크림	17.8 맑음		207
2025-01-01	수		0 서울	서울 강남역점	껌	17.8 맑음		189
2025-01-01	수		0 서울	서울 강남역점	에너지드링크	17.8 맑음		151
2025-01-02	목		0 서울	서울 강남역점	아이스 음료	20.8 맑음		173
2025-01-02	목		0 서울	서울 강남역점	생수	20.8 맑음		326
2025-01-02	목		0 서울	서울 강남역점	주류	20.8 맑음		149
2025-01-02	목		0 서울	서울 강남역점	마스크	20.8 맑음		140
2025-01-02	목		0 서울	서울 강남역점	컵라면	20.8 맑음		185
2025-01-02	목		0 서울	서울 강남역점	도시락	20.8 맑음		189
2025-01-02	목		0 서울	서울 강남역점	핫음료	20.8 맑음		177
2025-01-02	목		0 서울	서울 강남역점	아이스크림	20.8 맑음		208
2025-01-02	목		0 서울	서울 강남역점	껌	20.8 맑음		161
2025-01-02	목		0 서울	서울 강남역점	에너지드링크	20.8 맑음		199
2025-01-03	금		0 서울	서울 강남역점	아이스 음료	17.6 비		97
2025-01-03	금		0 서울	서울 강남역점	생수	17.6 비		203
2025-01-03	금		0 서울	서울 강남역점	주류	17.6 비		214
2025-01-03	금		0 서울	서울 강남역점	마스크	17.6 비		96
2025-01-03	금		0 서울	서울 강남역점	컵라면	17.6 비		310
2025-01-03	금		0 서울	서울 강남역점	도시락	17.6 비		143
2025-01-03	금		0 서울	서울 강남역점	핫음료	17.6 비		270

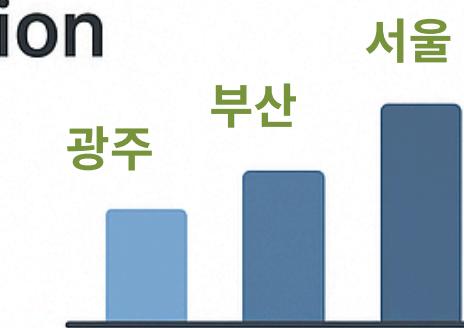
2025-12-29	월		0 옥천군	동하리점	도시락	15 맑음		9
2025-12-29	월		0 옥천군	동하리점	핫음료	15 맑음		19
2025-12-29	월		0 옥천군	동하리점	아이스크림	15 맑음		15
2025-12-29	월		0 옥천군	동하리점	껌	15 맑음		8
2025-12-29	월		0 옥천군	동하리점	에너지드링크	15 맑음		4
2025-12-30	화		0 옥천군	동하리점	아이스 음료	19.4 맑음		7
2025-12-30	화		0 옥천군	동하리점	생수	19.4 맑음		7
2025-12-30	화		0 옥천군	동하리점	주류	19.4 맑음		14
2025-12-30	화		0 옥천군	동하리점	마스크	19.4 맑음		7
2025-12-30	화		0 옥천군	동하리점	컵라면	19.4 맑음		19
2025-12-30	화		0 옥천군	동하리점	도시락	19.4 맑음		9
2025-12-30	화		0 옥천군	동하리점	핫음료	19.4 맑음		12
2025-12-30	화		0 옥천군	동하리점	아이스크림	19.4 맑음		10
2025-12-30	화		0 옥천군	동하리점	껌	19.4 맑음		8
2025-12-30	화		0 옥천군	동하리점	에너지드링크	19.4 맑음		4
2025-12-31	수		0 옥천군	동하리점	아이스 음료	19.7 맑음		10
2025-12-31	수		0 옥천군	동하리점	생수	19.7 맑음		8
2025-12-31	수		0 옥천군	동하리점	주류	19.7 맑음		12
2025-12-31	수		0 옥천군	동하리점	마스크	19.7 맑음		6
2025-12-31	수		0 옥천군	동하리점	컵라면	19.7 맑음		16
2025-12-31	수		0 옥천군	동하리점	도시락	19.7 맑음		6
2025-12-31	수		0 옥천군	동하리점	핫음료	19.7 맑음		12
2025-12-31	수		0 옥천군	동하리점	아이스크림	19.7 맑음		8
2025-12-31	수		0 옥천군	동하리점	껌	19.7 맑음		7
2025-12-31	수		0 옥천군	동하리점	에너지드링크	19.7 맑음		7

총 912,500개의 행으로 구성된 데이터셋

2. 데이터 탐색적 분석 (EDA)

**Day**

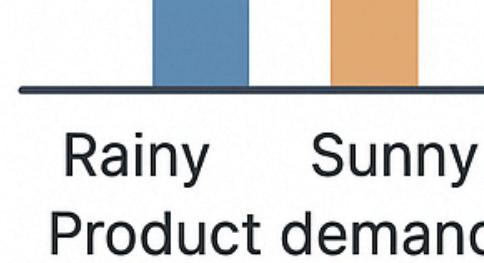
평일/주말 조건에 따른 물품 수요량 차이

**Region**

지역 조건에 따른 물품 수요량 차이

**Weather**

아이스 음료



날씨 조건에 따른 물품 수요량 차이

**Temperature**

아이스 음료



온도 조건에 따른 물품 수요량 차이

요일(평일/주말), 지역, 날씨, 온도 조건에 따라 물품의 수요량 차이가 발생 □ 학습 데이터 조건 충족

3. 모델 선택



Linear Regression

Baseline

- **선정 이유:** 모델 성능 평가의 기준점(Benchmark) 확보 및 전체적인 추세 파악.
- **특징:** 결과 해석이 직관적이며 과적합 위험이 낮음.



Random Forest

Stability

- **선정 이유:** 요일, 날씨 등 변수 간의 복잡한 **비선형 관계** 파악 및 변수 중요도 추출.
- **특징:** Bagging 방식을 통한 분산 감소로 안정적인 예측 성능 제공.



XGBoost

Accuracy

- **선정 이유:** 결측치(품절 등) 자동 처리 및 규제(Regularization)를 통한 고성능 예측.
- **특징:** Gradient Boosting 기반으로 오차를 순차적으로 개선하여 최고의 정확도 달성.



LightGBM

Efficiency

- **선정 이유:** 수많은 상품(SKU)과 점포 데이터를 처리하기 위한 **속도와 효율성**.
- **특징:** Leaf-wise 분할을 통해 빠른 학습 속도 제공, 메모리 효율이 높고 대규모 데이터에서도 안정적인 성능 확보 가능.

3. Linear Regression

머신러닝 학습 코드

```
df['날짜'] = pd.to_datetime(df['날짜'])
df['연'] = df['날짜'].dt.year
df['월'] = df['날짜'].dt.month
df['일'] = df['날짜'].dt.day
df['주차'] = df['날짜'].dt.isocalendar().week.astype(int)

df.drop(columns=['날짜'], inplace=True)
```

```
y = df['수요물품수']
X = df.drop(columns=['수요물품수'])
```

```
cat_features = ['요일', '지역', '매장', '물품', '날씨']
num_features = ['온도', '주말여부', '연', '월', '일', '주차']
```

```
X_encoded = pd.get_dummies(X, columns=cat_features, drop_first=False)
print("최종 Feature 개수:", X_encoded.shape[1])
```

최종 Feature 개수: 282

```
X_train, X_test, y_train, y_test = train_test_split(
    X_encoded, y,
    test_size=0.2,
    random_state=42
)
```

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
... ▾ LinearRegression ● ●
LinearRegression()
```

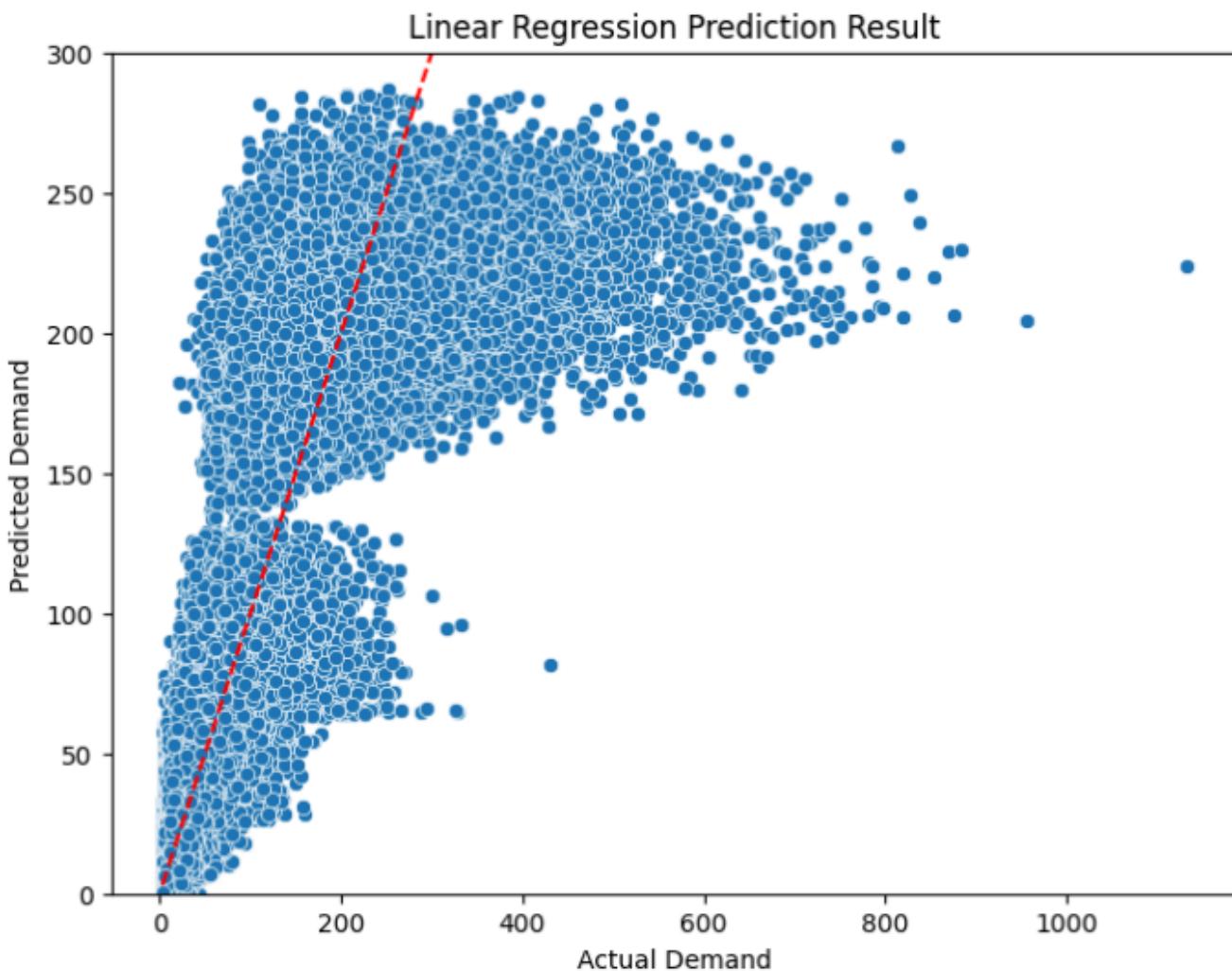
```
y_pred = model.predict(X_test)
```

```
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print("✓ Linear Regression 결과")
print("=====")
print(f"RMSE : {rmse:.2f}")
print(f"MAE : {mae:.2f}")
print(f"R² : {r2:.4f}")
```

```
✓ Linear Regression 결과
=====
RMSE : 51.30
MAE : 32.53
R² : 0.7370
```

시각화



전체적인 추세는 따라가지만, 특히 높은 수요 영역에서는 정확도가 떨어진다

3. Random Forest

머신러닝 학습 코드

```

# 입력(X), 타깃(y) 설정
X = df[['요일', '주말여부', '지역', '매장', '물품', '온도', '날씨']]
y = df['수요물품수']

# 범주형 변수(Label Encoding)
encoders = {}
for col in ['요일', '지역', '매장', '물품', '날씨']:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
    encoders[col] = le

#데이터 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestRegressor(
    n_estimators=200,          # 트리 수
    max_depth=15,             # 최대 깊이 제한
    min_samples_leaf=10,       # 최소 leaf 노드 샘플 수
    random_state=42,
    n_jobs=-1                 # 병렬 처리
)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

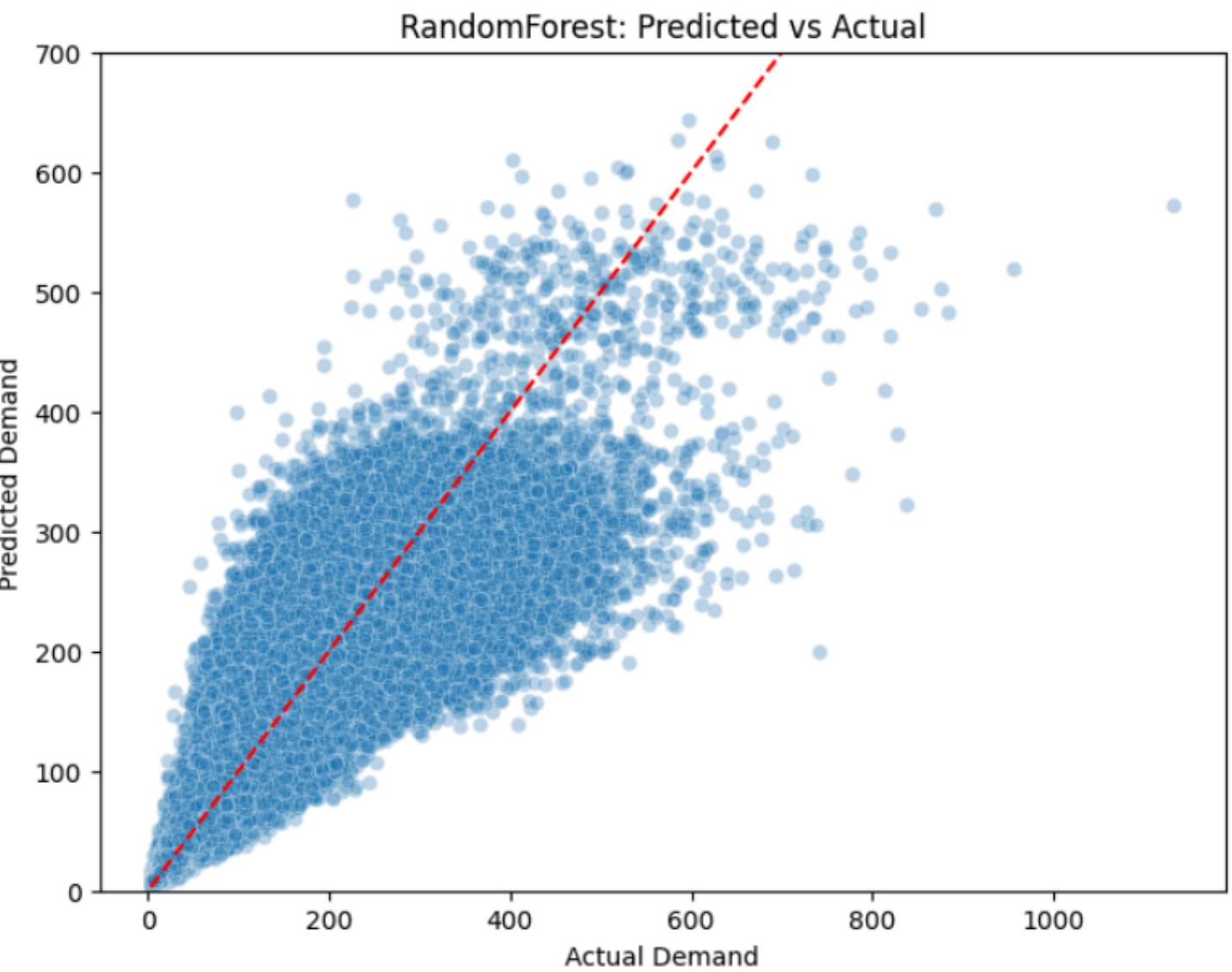
print(f"MAE : {mae:.2f}")
print(f"MSE : {mse:.2f}")
print(f"RMSE : {rmse:.2f}")
print(f"R2 : {r2:.4f}")

plt.figure(figsize=(8,6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.3)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.ylim(0,700)
plt.xlabel('Actual Demand')
plt.ylabel('Predicted Demand')
plt.title('RandomForest: Predicted vs Actual')
plt.show()

joblib.dump(model, 'rf_model.pkl')
joblib.dump(encoders, 'rf_label_encoders.pkl')

```

시각화



높은 수요 영역에서는 일부 예측 오차가 존재하여 극단 값에 대한 예측 정확도는 제한적

3. XGBoost

머신러닝 학습 코드

```
# 전처리 (Label Encoding)
# -----
cat_cols = ['요일', '지역', '매장', '물품', '날씨']
encoders = {}
for col in cat_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    encoders[col] = le

# feature / target 분리
features = ['요일', '주말여부', '지역', '매장', '물품', '온도', '날씨']
target = '수요물품수'

X = df[features]
y = df[target]

# 학습/검증 데이터 분리
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2, random_state=42)
dtrain = xgb.DMatrix(X_train, label=y_train)
dvalid = xgb.DMatrix(X_valid, label=y_valid)

params = {
    'objective': 'reg:squarederror',
    'learning_rate': 0.05,
    'max_depth': 6,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
    'seed': 42,
    'eval_metric': 'rmse'
}
evals = [(dtrain, 'train'), (dvalid, 'valid')]
num_round = 500
bst = xgb.train(
    params,
    dtrain,
    num_boost_round=num_round,
    evals=evals,
    early_stopping_rounds=50,
    verbose_eval=50
)

from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# 성능평가
# -----
y_pred = bst.predict(dvalid)
rmse = np.sqrt(mean_squared_error(y_valid, y_pred))
r2 = r2_score(y_valid, y_pred)
print(f"Validation RMSE: {rmse:.2f}")
print(f"Validation R^2 : {r2:.3f}")

# 오버피팅 확인
evals_result = {} # 학습/검증 기록 저장용

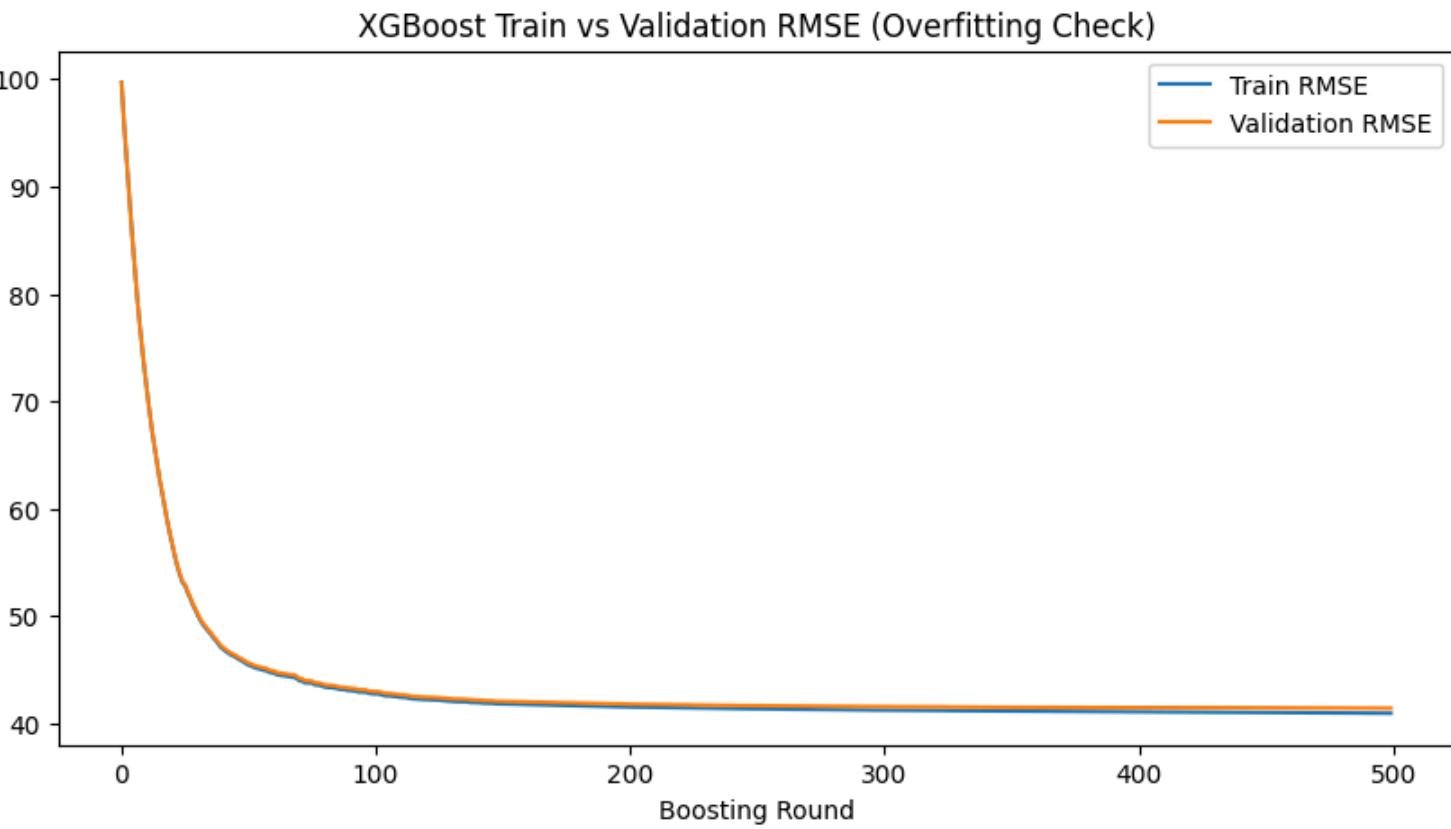
bst = xgb.train(
    params,
    dtrain,
    num_boost_round=num_round,
    evals=[(dtrain, 'train'), (dvalid, 'valid')],
    early_stopping_rounds=50,
    evals_result=evals_result, # 여기에 결과가 저장될
    verbose_eval=50
)

train_rmse = evals_result['train']['rmse']
valid_rmse = evals_result['valid']['rmse']

import matplotlib.pyplot as plt

plt.figure(figsize=(10,5))
plt.plot(train_rmse, label='Train RMSE')
plt.plot(valid_rmse, label='Validation RMSE')
plt.xlabel('Boosting Round')
plt.ylabel('RMSE')
plt.title('XGBoost Train vs Validation RMSE (Overfitting Check)')
plt.legend()
plt.show()
```

시각화



RMSE가 거의 동일하게 유지되는 것을 확인
일반화 성능이 우수하며 학습 과정이 안정적임

3. LightGBM

머신러닝 학습 코드

```

# ① LightGBM 모델 학습

# ① 전처리 (Label Encoding)
# 데이터 로드
df = pd.read_excel("2total_daily_data.xlsx", engine="openpyxl")

# Feature / Target 정의
features = ["요일", "주말여부", "지역", "매장", "물품", "온도", "날씨"]
target = "수요물품수"

df = df[features + [target]].copy()

# Label Encoding
cat_cols = ["요일", "주말여부", "지역", "매장", "물품", "날씨"]
label_encoders = {}

for col in cat_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# ② 학습 / 검증 데이터 분리

X = df[features]
y = df[target]

X_train, X_valid, y_train, y_valid = train_test_split(
    X, y, test_size=0.2, random_state=42, shuffle=True
)

train_data = lgb.Dataset(X_train, label=y_train)
valid_data = lgb.Dataset(X_valid, label=y_valid)

# ③ 성능 평가

y_pred = model.predict(X_valid)

mae = mean_absolute_error(y_valid, y_pred)
rmse = np.sqrt(mean_squared_error(y_valid, y_pred))
r2 = r2_score(y_valid, y_pred)

print("===== LightGBM 성능 =====")
print(f"MAE : {mae:.3f}")
print(f"RMSE : {rmse:.3f}")
print(f"R2 : {r2:.3f}")

# ④ 오버피팅 확인 (Train/Valid RMSE 시각화)

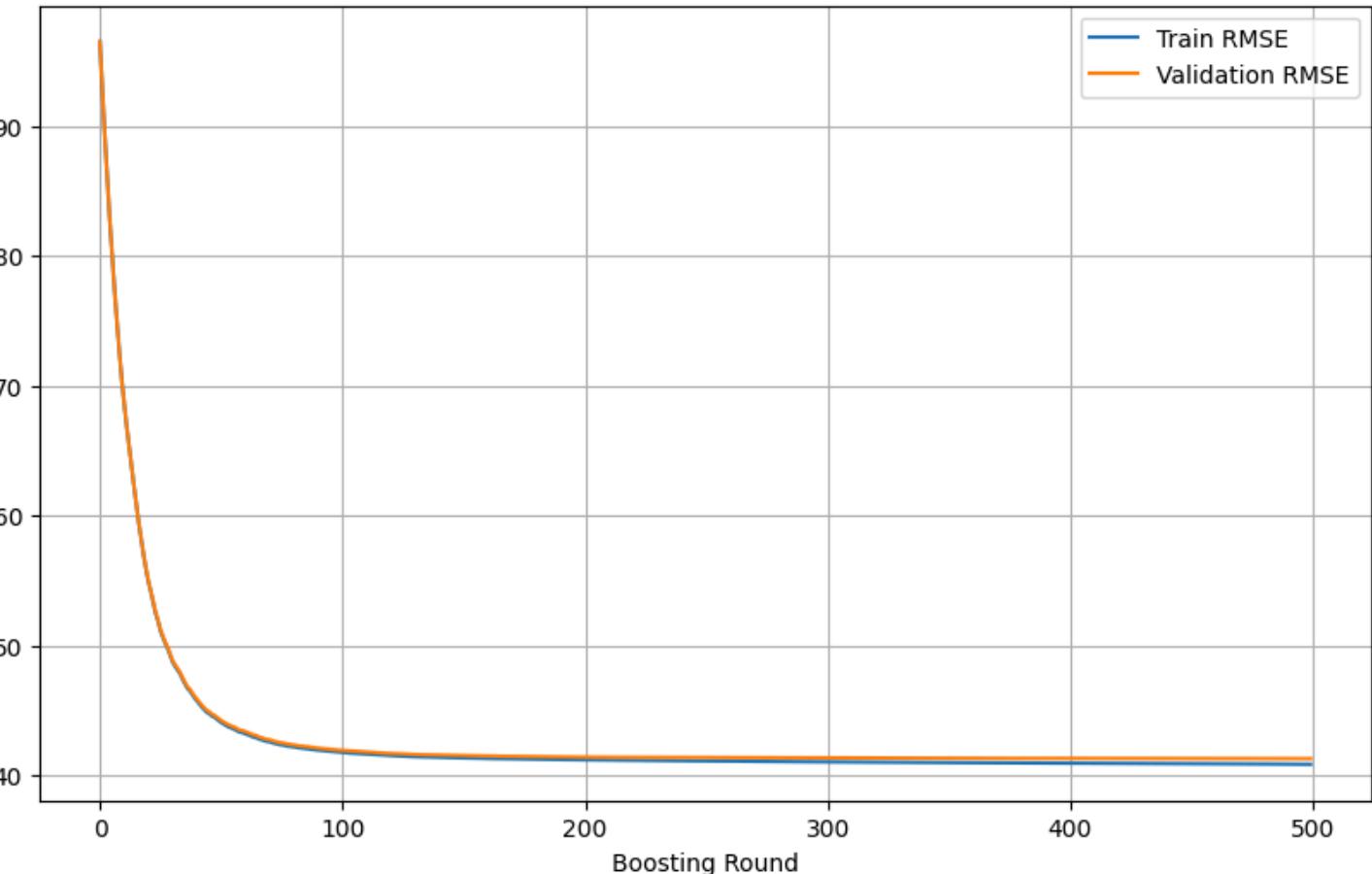
train_rmse = evals_result["train"]["rmse"]
valid_rmse = evals_result["valid"]["rmse"]

plt.figure(figsize=(10, 6))
plt.plot(train_rmse, label="Train RMSE")
plt.plot(valid_rmse, label="Validation RMSE")
plt.title("LightGBM Train vs Validation RMSE (Overfitting Check)")
plt.xlabel("Boosting Round")
plt.ylabel("RMSE")
plt.legend()
plt.grid(True)
plt.show()

```

시각화

LightGBM Train vs Validation RMSE (Overfitting Check)



RMSE가 거의 동일하게 유지되는 것을 확인
일반화 성능이 우수하며 학습 과정이 안정적임

3. 모델 평가

	Linear Regression	Random Forest	XGBoost	LightGBM
처리 속도	빠름	매우 느림	빠름	느림
예측 정확도	부족	우수	우수	우수
과적합 위험	낮음	중간	높음 (과적합 여부 확인)	높음 (과적합 여부 확인)
성능(R^2)	0.73	0.82	0.83	0.83

4. 구현 방향

예상 수요 예측 서비스의 주요 기능 정의

⚡ 핵심 예측 변수 선택 (Input Selection)

사용자가 **요일, 지역, 매장, 물품, 온도 구간, 날씨 조건** 6가지 주요 조건을 선택하여 머신러닝 모델에 입력하고, 이에 기반한 최적의 수요 예측 결과를 제공합니다.

1. 요일 조건

2. 지역 선택

3. 매장 구분

4. 물품 종류

5. 온도 구간

6. 날씨 상황

최종 목표 (Output Target)

예상 수요 확인

가장 높은 정확도의 모델이 예측한
최종 수요량 제공

- ✓ 사용자 맞춤형 시뮬레이션 환경 제공
- ✓ 빠른 의사결정을 위한 직관적인 UI/UX
- ✓ 6대 변수 영향 분석 기능

4. Streamlit 코드

캐시 및 파일 로드

```
# -----  
# ❶ 캐시된 함수  
# -----  
@st.cache_data  
def load_excel(path):  
    return pd.read_excel(path)  
  
@st.cache_resource  
def load_xgb_model(path):  
    model = xgb.Booster()  
    model.load_model(path)  
    return model  
  
@st.cache_resource  
def load_lgbm_model(path):  
    return joblib.load(path)  
  
@st.cache_resource  
def load_joblib_model(path):  
    return joblib.load(path)  
  
@st.cache_resource  
def load_encoders(path):  
    return joblib.load(path)  
  
# -----  
# ❷ 파일 경로 / 파일명  
# -----  
EXCEL_PATH = "2total_daily_data.xlsx"  
  
# XGBoost  
XGB_MODEL_PATH = "xgboost_model.json"  
XGB_ENCODER_PATH = "label_encoders.joblib"  
  
# Linear Regression  
LINEAR_MODEL_PATH = "linear_model.pkl"  
LINEAR_COLS_PATH = "linear_model_columns.pkl"  
LINEAR_ENCODER_PATH = XGB_ENCODER_PATH # 기존 XGB 인코더 사용  
  
# Random Forest  
RF_MODEL_PATH = "rf_model_compressed.pkl"  
RF_ENCODER_PATH = "rf_label_encoders.pkl"  
  
# LightGBM  
LGBM_MODEL_PATH = "future_lgbm_model.pkl"  
LGBM_ENCODER_PATH = "future_label_encoders.pkl"
```

데이터 및 모델 로드

Streamlit 코드

```
# -----  
# ❸ Streamlit UI  
# -----  
st.title("➊ 편의점 수요 예측 시스템")  
  
# 모델 선택  
model_name = st.selectbox("모델 선택", list(models_dict.keys()))  
  
# 요일 선택 → 주말 여부 자동 계산  
요일_list = ["월", "화", "수", "목", "금", "토", "일"]  
selected_day = st.selectbox("요일 선택", 요일_list)  
주말여부 = 1 if selected_day in ["토", "일"] else 0  
st.text(f"주말 여부: {'주말' if 주말여부==1 else '평일'}")  
  
# 지역 선택  
regions = df['지역'].unique().tolist()  
selected_region = st.selectbox("지역 선택", regions)  
  
# 선택한 지역 매장 선택  
region_stores = df[df['지역']==selected_region]['매장'].unique().tolist()  
selected_store = st.selectbox("매장 선택", region_stores)  
  
# 물품 선택  
items = df['물품'].unique().tolist()  
selected_item = st.selectbox("물품 선택", items)  
  
# 온도 선택  
temp_list = ["-10~0°C", "1~10°C", "11~20°C", "21~30°C", "31~40°C"]  
selected_temp = st.selectbox("평균 기온 구간", temp_list)  
temp_mapping = {"-10~0°C":0, "1~10°C":5, "11~20°C":15, "21~30°C":25, "31~40°C":35}  
온도_val = temp_mapping[selected_temp]  
  
# 날씨 선택  
weather_list = df['날씨'].unique().tolist()  
selected_weather = st.selectbox("날씨 선택", weather_list)
```

모델 예측 코드

```
# -----  
# 모델별 예측  
# -----  
if model_name == "XGBoost":  
    encode_cols = ["요일", "지역", "매장", "물품", "날씨"]  
    for col in encode_cols:  
        encoder = xgb_encoders[col]  
        input_df[col] = safe_transform(encoder, input_df[col].astype(str))  
    dmatrix = xgb.DMatrix(input_df)  
    pred = models_dict[model_name].predict(dmatrix)[0]  
  
elif model_name == "Linear Regression":  
    input_df = input_df.reindex(columns=linear_cols, fill_value=0)  
    pred = models_dict[model_name].predict(input_df)[0]  
  
elif model_name == "Random Forest":  
    encode_cols = ["요일", "지역", "매장", "물품", "날씨"]  
    for col in encode_cols:  
        encoder = rf_encoders[col]  
        input_df[col] = safe_transform(encoder, input_df[col].astype(str))  
    pred = models_dict[model_name].predict(input_df)[0]  
  
elif model_name == "LightGBM":  
    encode_cols = ["요일", "지역", "매장", "물품", "날씨"]  
    for col in encode_cols:  
        encoder = lgbm_encoders[col]  
        input_df[col] = safe_transform(encoder, input_df[col].astype(str))  
    pred = models_dict[model_name].predict(input_df)[0]  
  
예측수요 = np.round(pred)  
권장발주량 = np.round(pred * 1.1)  
  
# -----  
# ❹ 결과 출력  
# -----  
st.subheader("❻ 예측 결과")  
st.write(f"선택 모델: {model_name}")  
st.write(f"예측 수요: {예측수요}")  
st.write(f"권장 발주량 (10% 여유): {권장발주량}")  
  
# -----  
# ❺ 과거 조건과 비교 그래프  
# -----  
filtered_df = df[  
    (df['요일'] == selected_day) &  
    (df['지역'] == selected_region) &
```

4. UI 구현

📦 편의점 수요 예측 시스템

모델 선택

XGBoost

요일 선택

월

주말 여부: 평일

지역 선택

서울

매장 선택

서울 강남역점

물품 선택

아이스 음료

평균 기온 구간

-10~0°C

날씨 선택

맑음

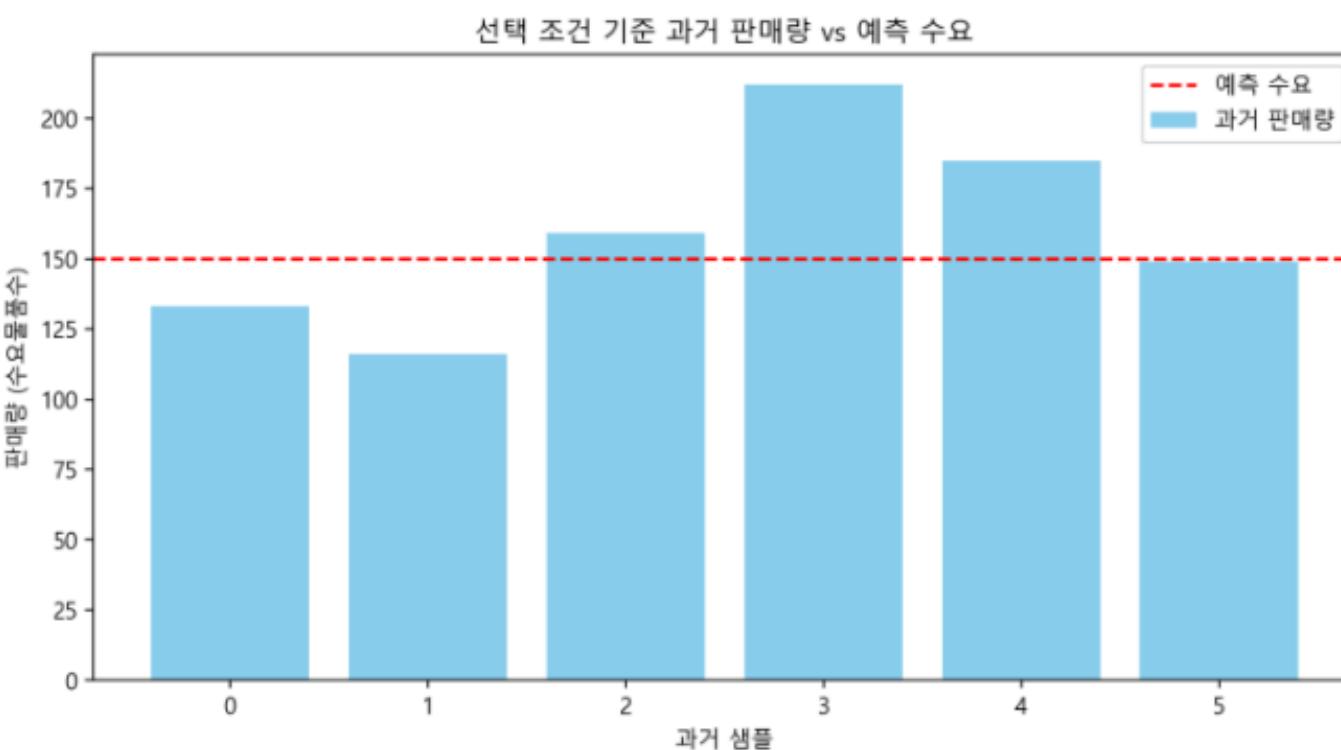
예상 수요 확인

예측 결과

선택 모델: XGBoost

예측 수요: 150.0

권장 발주량 (10% 여유): 165.0



- 월요일은 평일이라 평균적인 수요 수준입니다.
- 선택된 지역/매장(서울 / 서울 강남역점)의 과거 판매 패턴을 반영했습니다.
- 물품(아이스 음료)은 과거 데이터 기준으로 평균 판매량이 계산되었습니다.
- 날씨(맑음)와 온도(-10~0°C)에 따라 구매량에 영향을 줄 수 있습니다. → 모델은 입력 조건과 가장 유사한 과거 데이터를 기반으로 예측했습니다.

5. 기대효과

핵심 성과 영역 (Key Impact Areas)



고정밀 예측 기반 발주 의사결정

머신러닝 기반 고정밀 예측(83% 이상 목표)을 통해 점주/물류 담당자의 발주량 결정에 객관적 지표를 제공



매출 및 효율 증대

발주 물류 최적화를 통한 폐기 비용 최소화 및 품절 방지로 고객 만족도 증대와 수익 증대 효과를 동시에 달성



발주 물류 최적화 시스템 기틀

50개 매장 데이터를 기반으로 재고, 물류, 발주 시스템을 통합할 수 있는 데이터 중심 관리 체계의 핵심 기반을 마련

5. 자체 피드백

프로젝트 구현의 주요 한계점 (Current Limitations)

! 1. 독립 변수(Feature)의 예측력 한계

현재 모델은 요일, 지역, 매장, 물품, 온도, 날씨 조건 등 6가지 기본 변수에만 의존하고 있습니다. 이는 실제 수요를 예측하는 데 필요한 매우 많은 외부 및 내부 변수(경쟁사, 고객 선호도, 시즌성 등)에 비해 부족하여 예측 성능 개선에 한계를 보였습니다.

! 2. 이벤트/프로모션 데이터의 부재

실제 할인, 1+1, 특정 날짜 이벤트 등과 같은 판매 촉진 이벤트 데이터가 확보되지 않아, 단기적인 수요 급증 패턴을 예측 모델에 반영하지 못했습니다. 이는 발주량 추천의 정확성을 떨어뜨리는 주요 원인입니다.

향후 개선 방향 및 모델 고도화 로드맵

 유효 Feature 추가 탐색 및 확보하여(상권 특성, 경쟁 환경, 유동 인구 등 유효성이 입증된 추가 독립 변수)기존 모델의 예측 정확도를 획기적으로 향상

 이벤트 데이터(물품 행사, 날짜 이벤트 등 내부 판매 데이터)를 체계적으로 수집하고 모델에 반영하여, 이벤트 기간의 수요 변화를 고정밀로 예측가능하게 하는 파이프라인 구축

| 감사합니다.

Supporting data

모델 선택

XGBoost

XGBoost

Linear Regression

Random Forest

LightGBM

요일 선택

월

월

화

수

지역 선택

서울

서울

경기도

부산

광주

옥천군

매장 선택

서울 강남역점

서울 강남역점

서울 역삼역점

서울 삼성역점

서울 선릉역점

서울 서초역점

서울 잠실새내점

서울 잠실역점

서울 소공그쳐져

물품 선택

아이스 음료

아이스 음료

생수

주류

마스크

컵라면

도시락

핫음료

아이스크리

평균 기온 구간

-10~0°C

-10~0°C

1~10°C

11~20°C

21~30°C

31~40°C

날씨 선택

맑음

맑음

비

황사

눈