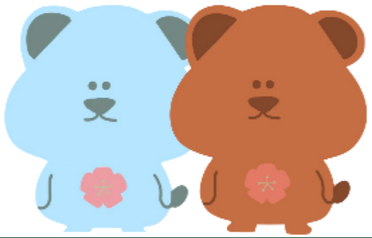
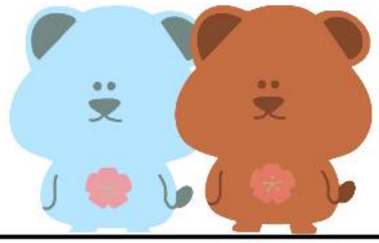
데이콘 Basic
쇼핑몰 지점별 매출액 예측 경진대회

이수연



EWHA EURON

목차



EWHA EURON

#1-1. 대회 & 데이터 소개

목표: 쇼핑몰의 프로모션, 주변 기온, 휴일 정보 등을 이용한 지점별 한 달 매출액 예측

평가지표: $\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}}$

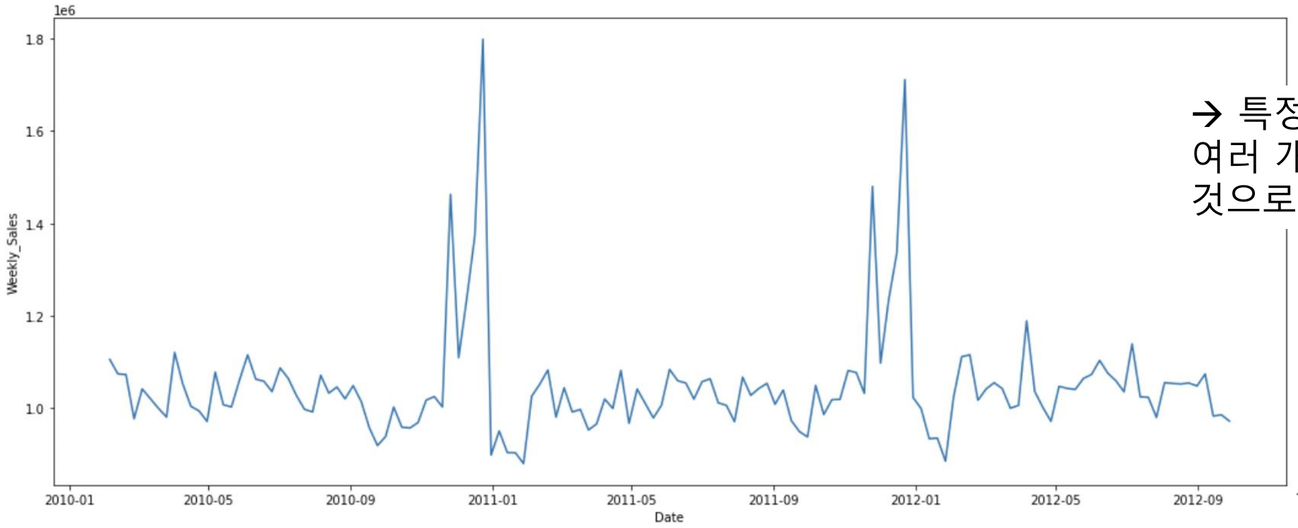
Features

- id : 샘플 아이디
- Store : 쇼핑몰 지점
- Date : 주 단위(Weekly) 날짜
- Temperature : 해당 쇼핑몰 주변 기온
- Fuel_Price : 해당 쇼핑몰 주변 연료 가격
- Promotion 1~5 : 해당 쇼핑몰의 비식별화된 프로모션 정보
- Unemployment : 해당 쇼핑몰 지역의 실업률
- IsHoliday : 해당 기간의 공휴일 포함 여부
- Weekly_Sales : 주간 매출액 (목표 예측값!)

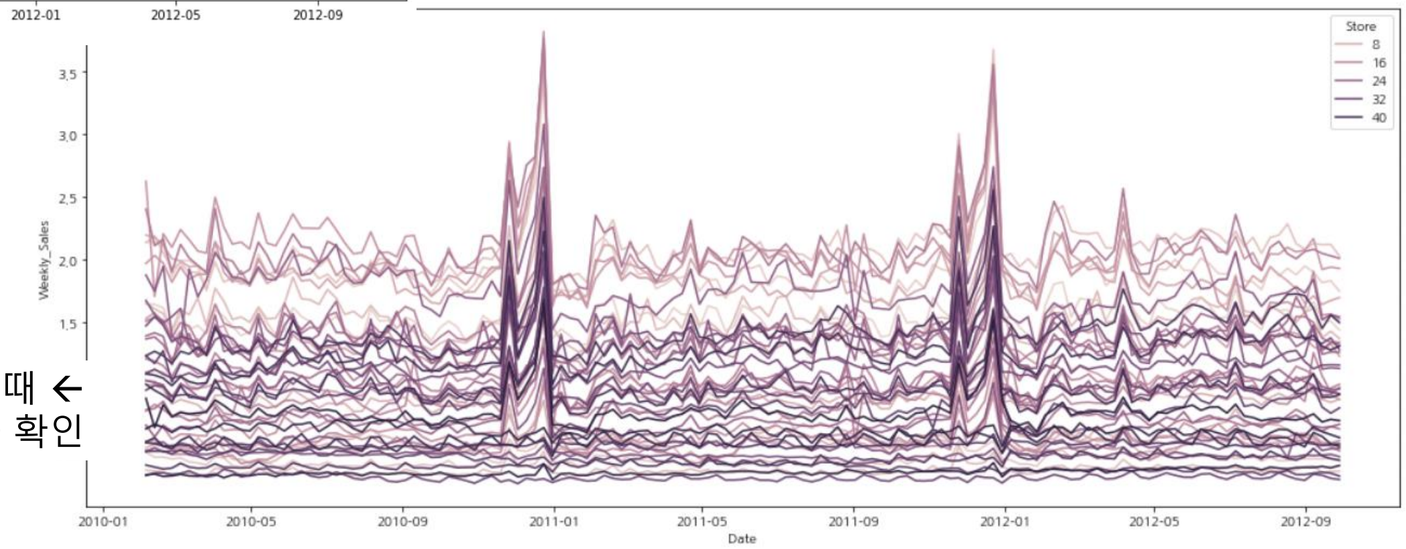
	id	Store	Date	Temperature	Fuel_Price	Promotion1	Pro...	Pro...	Pro...	Prom...	Unemployment	IsHoliday	Weekly_Sales
1	1	1	05/02/2010	42.31	2.572						8.106	False	1643690.90
2	2	1	12/02/2010	38.51	2.548						8.106	True	1641957.44
3	3	1	19/02/2010	39.93	2.514						8.106	False	1611968.17
4	4	1	26/02/2010	46.63	2.561						8.106	False	1409727.59
5	5	1	05/03/2010	46.50	2.625						8.106	False	1554806.68

#1-1. 대회 & 데이터 소개

데이터 소개 - Date



→ 특정 기간에 매출이 급격히 오르고 내리는 경우가 있음.
여러 개의 매장이 합쳐져 있는 값이므로 극단값의 영향이 클 것으로 예측



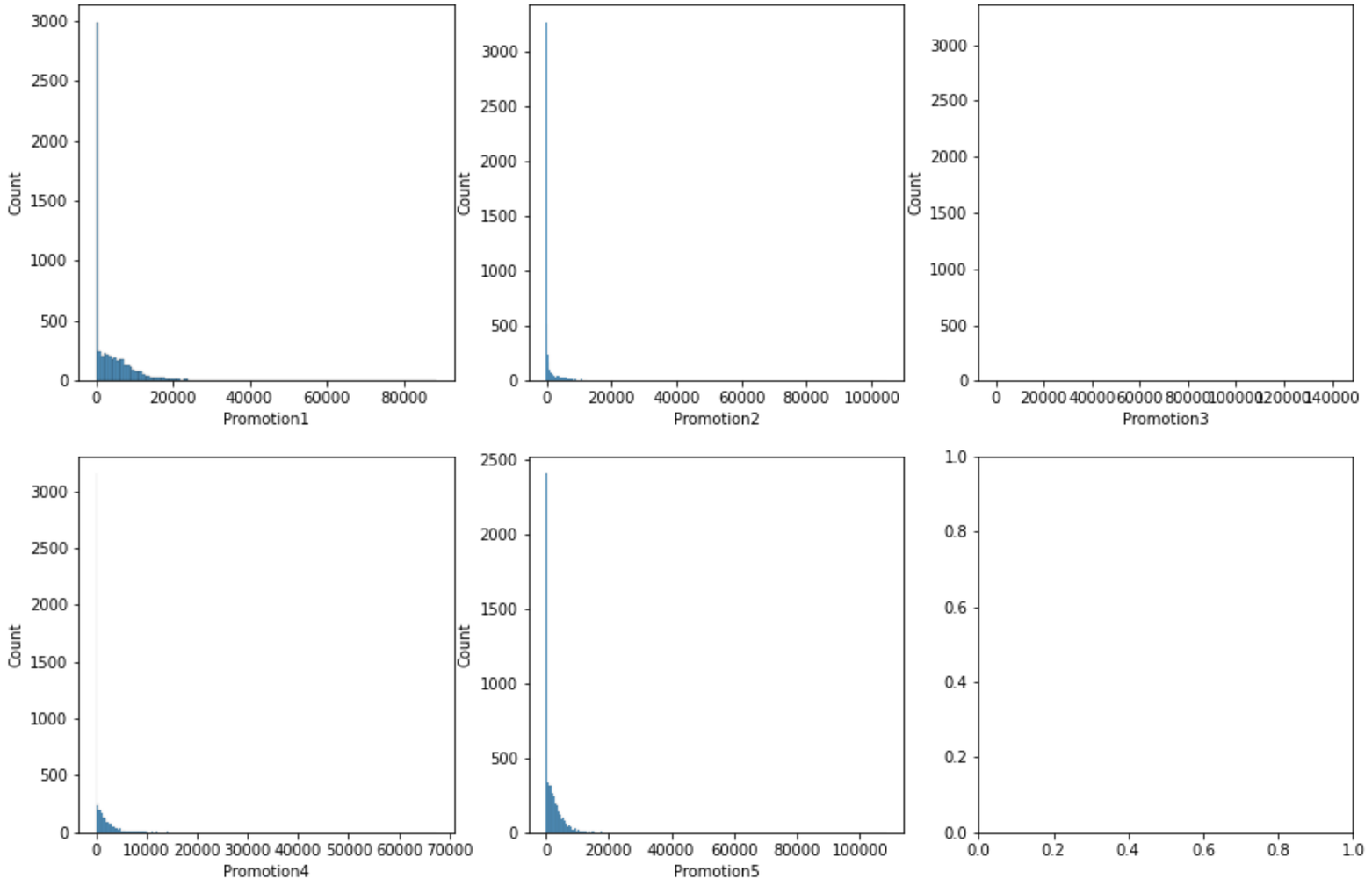
Store별로 나누어 보았을 때 ←
특정 매장의 매출이 급격하게 변하는 시점이 있는 것을 확인

#1-1. 대회 & 데이터 소개

데이터 소개 - promotion

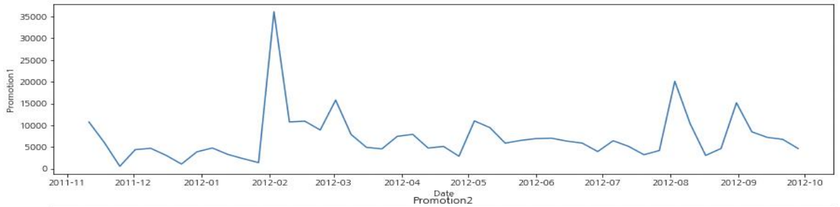
```
Promotion1: 4153
Promotion2: 4663
Promotion3: 4370
Promotion4: 4436
Promotion5: 4140
---------
Promotion1: 2
Promotion2: 135
Promotion3: 19
Promotion4: 34
Promotion5: 0
```

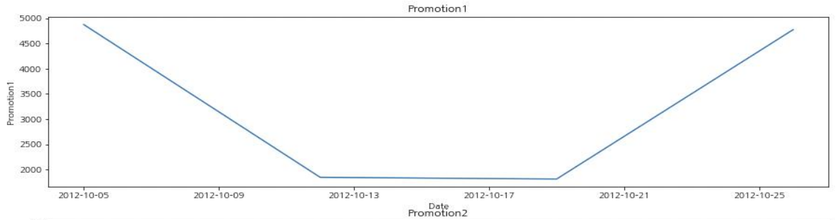
→ 2/3 가량이 결측값

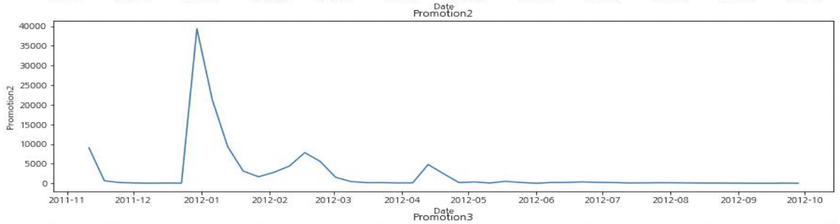


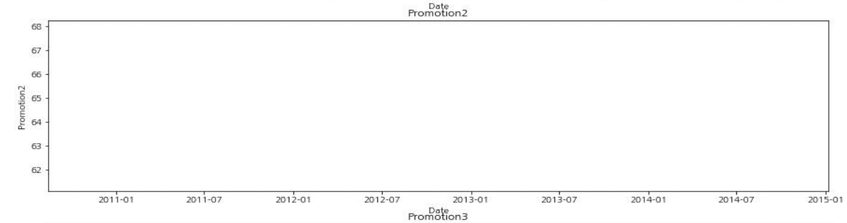
#1-1. 대회 & 데이터 소개

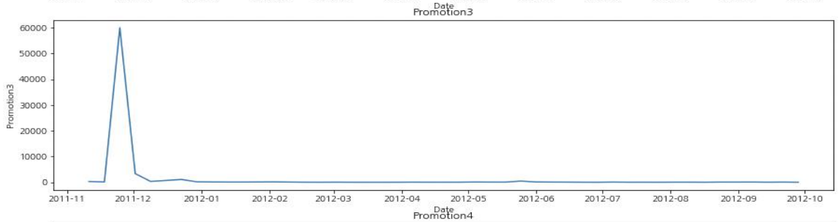
데이터 소개 - promotion

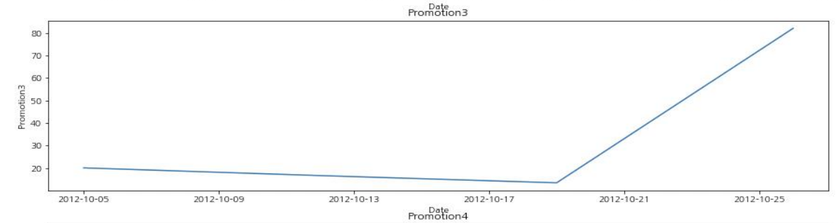


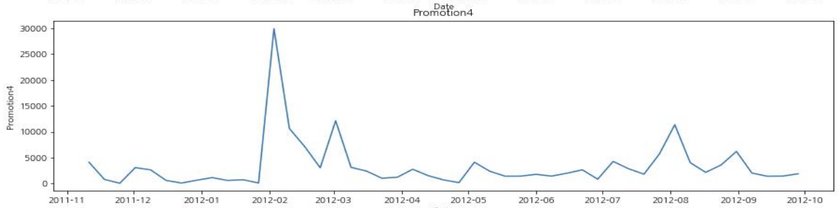


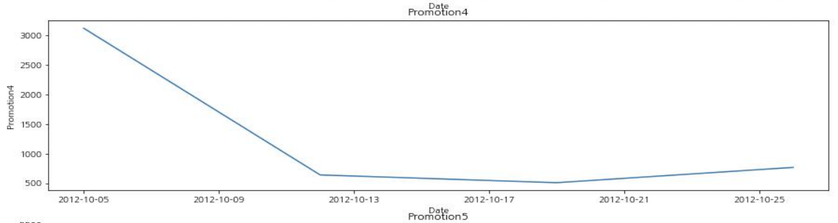


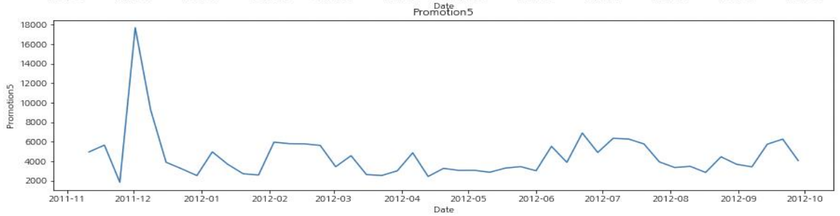


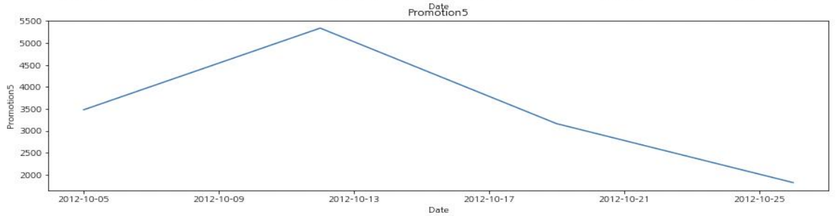












→ Promotion이 2011년 11월 이후 발생한 이벤트라고 생각할 수 있음.

#1-2. 데이터 전처리

promotion은 2011년 11월 이후 발생한 이벤트로,
2011년 11월 이전의 promotion 결측치는 0으로 처리하고, 해당하지 않는 데이터는 선형 보간법을 이용.

```
def fill_promotion(cols):
  year, promotion = cols[0], cols[1]
  if pd.isnull(promotion):
    if year<2011:
      return 0
    else:
      return promotion
  else:
    return promotion
```

```
for p in promotion:
  temp[p] = temp[['Year', p]].apply(fill_promotion, axis=1)
```

```
# 선형 보간
temp[promotion] = temp[promotion].interpolate("values")
```

	Promotion1	Promotion2	Promotion3	Promotion4	Promotion5
0	0.00	0.00	0.00	0.00	0.00
1	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
...	...	...	...	...	...
175	38.65	2.61	0.98	0.00	457.74
176	5046.74	0.00	18.82	2253.43	2340.01
177	1956.28	0.00	7.89	599.32	3990.54
178	2004.02	0.00	3.18	437.73	1537.49
179	4018.91	58.08	100.00	211.94	858.33

6435 rows × 5 columns

#2-1. Pycaret - setup

```
import jinja2
from pycaret.regression import *
exp = setup(data = train, target = 'Weekly_Sales',session_id=42,  normalize = True, remove_outliers= True,  remove_multicollinearity = True,
            ignore_low_variance = True)
```

	Description	Value
0	session_id	42
1	Target	Weekly_Sales
2	Original Data	(6255, 13)
3	Missing Values	False
4	Numeric Features	8
5	Categorical Features	4
6	Ordinal Features	False
7	High Cardinality Features	False
8	High Cardinality Method	None
9	Transformed Train Set	(4159, 27)
10	Transformed Test Set	(1877, 27)
11	Shuffle Train-Test	True
12	Stratify Train-Test	False
13	Fold Generator	KFold

```
INFO:logs:create_model_container: 0
INFO:logs:master_model_container: 0
INFO:logs:display_container: 1
INFO:logs:Pipeline(memory=None,
         steps=[('dtypes',
                 DataTypes_Auto_infer(categorical_features=[],
                                      display_types=True, features_todrop=[],
                                      id_columns=[], ml_usecase='regression',
                                      numerical_features=[],
                                      target='Weekly_Sales',
                                      time_features=[])),
                ('imputer',
                 Simple_Imputer(categorical_strategy='not_available',
                                fill_value_categorical=None,
                                fill_value_numerical=None,
                                numeric_str...
                ('dummy', Dummify(target='Weekly_Sales')),
                ('fix_perfect', Remove_100(target='Weekly_Sales')),
                ('clean_names', Clean_Colum_Names()),
                ('feature_select', 'passthrough'),
                ('fix_multi',
                 Fix_multicollinearity(correlation_with_target_preference=None,
                                       correlation_with_target_threshold=0.0,
                                       target_variable='Weekly_Sales',
                                       threshold=0.9)),
                ('dfs', 'passthrough'), ('pca', 'passthrough')],
         verbose=False)
INFO:logs:setup() succesfully completed.....................................
```

#2-2. Pycaret – compare, create model

compare_models

```
best = compare_models(include = ['lightgbm','rf'], sort='RMSE')
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
lightgbm	Light Gradient Boosting Machine	302237.1800	1.650060e+11	405985.4002	0.4675	0.4265	0.3895	0.462
rf	Random Forest Regressor	313861.8455	1.992842e+11	446131.5572	0.3567	0.4560	0.4010	2.261

create_models

```
lgbm = create_model('lightgbm', cross_validation = False)
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	314709.7891	1.857097e+11	430940.5025	0.4305	0.4402	0.4035

#2-3. Pycaret – Predict

```
predict_model(lgbm, data=test)
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Light Gradient Boosting Machine	0	0	0	0	0	0

	id	Store	Date	Temperature	Fuel_Price	Promotion1	Promotion2	Promotion3	Promotion4	Promotion5	Unemployment	IsHoliday	Weekly_Sales	Year	Month	Day	Label
0	1	1	2012-10-05	68.55	3.617	8077.89	0.00	18.22	3617.43	3626.14	Low	False	NaN	2012	2	5	1.586825e+06
1	2	1	2012-10-12	62.99	3.601	2086.18	0.00	8.11	602.36	5926.45	Low	False	NaN	2012	2	12	1.394646e+06
2	3	1	2012-10-19	67.97	3.594	950.33	0.00	4.93	80.25	2312.85	Low	False	NaN	2012	2	19	1.177481e+06
3	4	1	2012-10-26	69.16	3.506	2585.85	31.75	6.00	1057.16	1305.01	Low	False	NaN	2012	2	26	1.100684e+06
4	5	2	2012-10-05	70.27	3.617	6037.76	0.00	10.04	3027.37	3853.40	Low	False	NaN	2012	3	5	1.847018e+06
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
175	176	44	2012-10-26	46.97	3.755	38.65	2.61	0.98	0.00	457.74	Low	False	NaN	2012	10	26	2.861399e+05
176	177	45	2012-10-05	64.89	3.985	5046.74	0.00	18.82	2253.43	2340.01	Middle	False	NaN	2012	10	5	7.574877e+05
177	178	45	2012-10-12	54.47	4.000	1956.28	0.00	7.89	599.32	3990.54	Middle	False	NaN	2012	10	12	7.716495e+05
178	179	45	2012-10-19	56.47	3.969	2004.02	0.00	3.18	437.73	1537.49	Middle	False	NaN	2012	11	19	7.507014e+05
179	180	45	2012-10-26	58.85	3.882	4018.91	58.08	100.00	211.94	858.33	Middle	False	NaN	2012	11	26	7.939391e+05

180 rows × 17 columns

#3-1. 1등 솔루션 - Promotion 전처리

1. 음수값 처리

- Promotion에 대한 해석이 다양하게 존재할 수 있고,
  음수값을 처리할만한 충분한 근거를 찾지 못해 그대로 둠.

- 이후 feature selection에서 promotion2가 사용되지 않았으므로 음수값
  처리가 큰 영향을 미치지는 않을 것으로 판단.

2. 결측치 처리

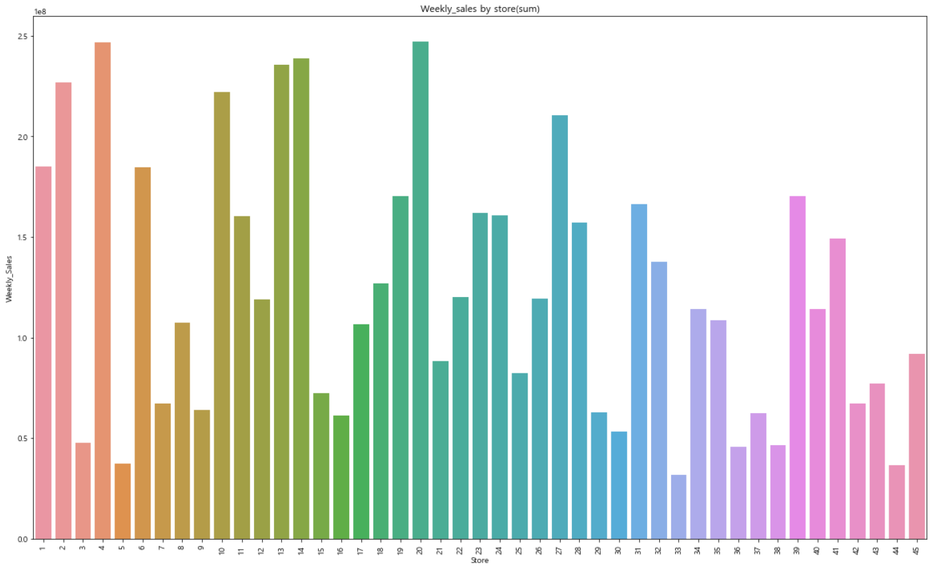
1. 2011년 11월 이전에는 promotion이 진행되지 않았다고 판단.
2. 2011년 11월 이전에도 promotion이 진행되었지만 기록이 이후부터 되었다고 판단.

→45개의 지점 모두가 동일한 날에 promotion을 시작했다고 보기는 어렵다고 판단해 2번 방법을 조금 더 신뢰할 수 있다고 판단.

→이전의 결측값은 0으로 채워주고 이후는 선형보간법을 채워주려고 함

→결론적으로 선형 보간법보다 모두 0으로 채운 모델의 성능이 더 높았음.

#3-1. 1등 솔루션



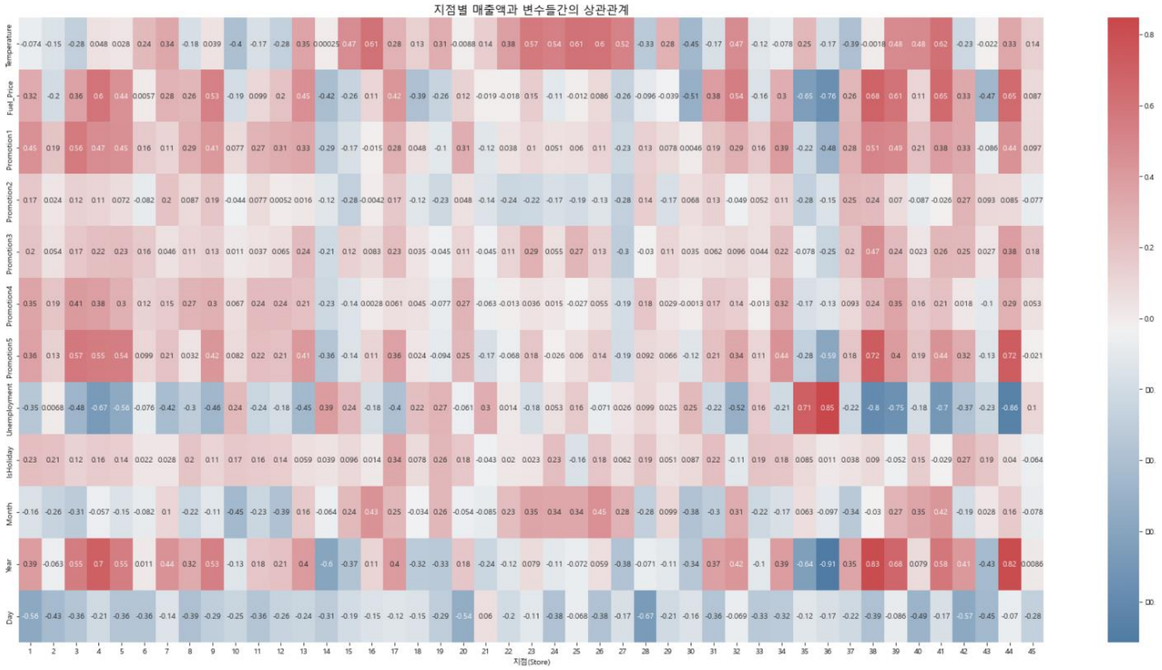
지점별로 매출액의 차이가 크므로
지점별 모델을 만들어 따로 예측.

```
features = ['IsHoliday','Month','Year','Day']
features_1_3_5 = ['IsHoliday','Month','Year','Day','Promotion1','Promotion3','Promotion5']
features_1_4_5 = ['IsHoliday','Month','Year','Day','Promotion1','Promotion4','Promotion5']
features_1_5 = ['IsHoliday','Month','Year','Day','Promotion1','Promotion5']
features_1 = ['IsHoliday','Month','Year','Day','Promotion1']
features_5 = ['IsHoliday','Month','Year','Day','Promotion5']
```

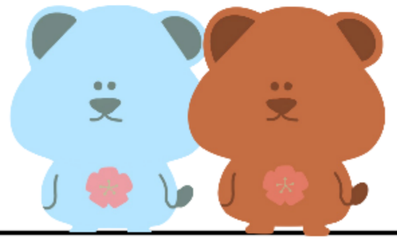
CatBoost 사용
→ store 별로 모델을 따로 만들어서 예측하다보니
ordered boosting 방식을 사용하는 catboost의 성능이 좋았을 것.

Feature selection

지점별 매출액과 변수간의 상관관계 확인.
→ Store별로 promotion간 상관관계가 0.4 이상인 feature만 선택.
→ Promotion1, promotion5 만 사용
→temperature, fuel_price, unemployment 성능에 방해



THANK YOU



EWHA EURON