# NS SHOP+ 판매실적 예측을 통한 편성 최적화 방안(모형) 도출

팀명 : 포벤져스
팀원 : 성수호, 최형찬, 한충완

# 목차

# 1. 방송 편성표에 따른 판매실적 예측

## 결측치 확인 및 처리

### 결측치 확인

```
방송일시          0
노출(분)      16784
마더코드          0
상품코드          0
상품명           0
상품군           0
판매단가          0
 취급액        2930
dtype: int64
```

>> '노출(분)', '취급 액' 열에서 결측 치 발견

### 결측치 처리

'노출(분)'

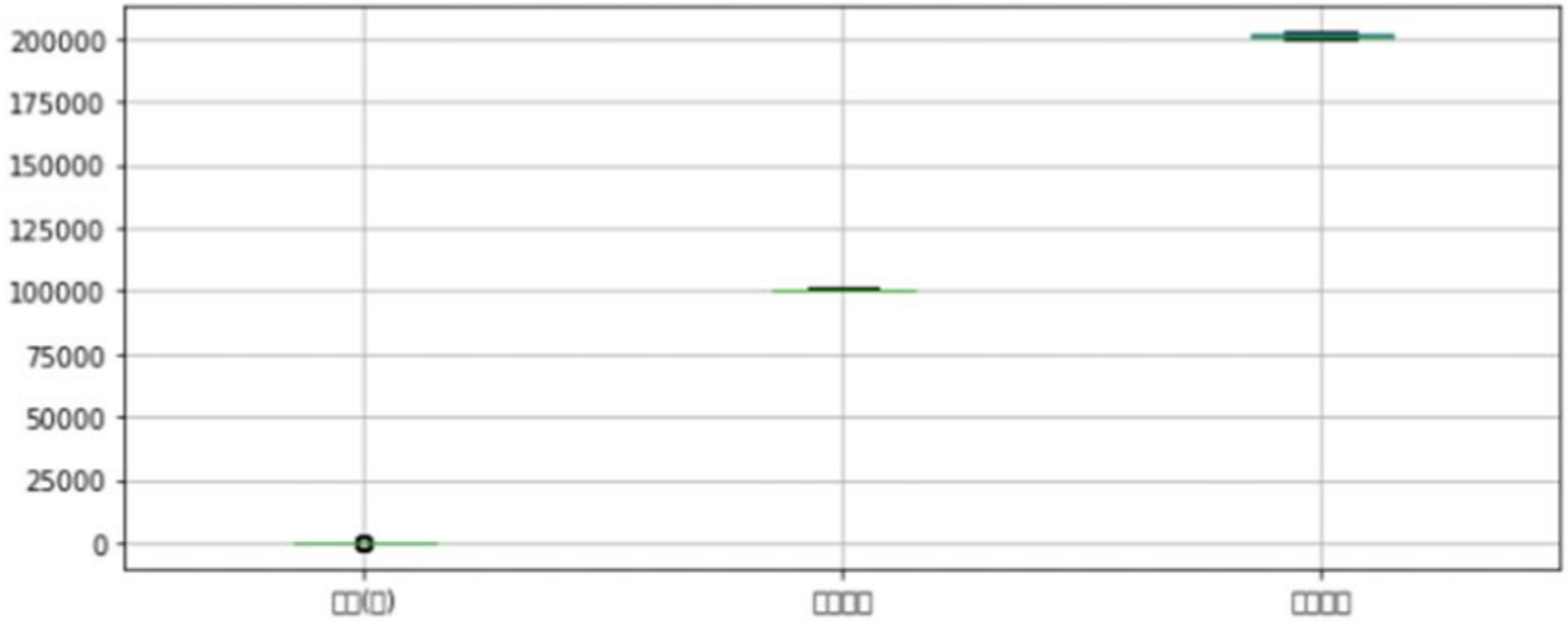
'노출(분)' 에 대한 결측 값은 바로 앞 행과 동 시간대이므로

==> 바로 앞 행 값으로 결측 값 채움.

'취급액'

결측 값 있는 행 전체 삭제

## 이상치 확인 및 처리




>> 전체 열에 대해서 이상치 확인 한 결과 이상치 없음

## 컬럼 별 정리

### '판매단가','취급액'

```
df_raw['판매단가']= df_raw['판매단가'].replace(',','', regex=True).astype(int)

df_raw[' 취급액 ']= df_raw[' 취급액 '].replace(',','', regex=True).astype(int)
```

타입 변환 : object ==> int 로 변환

### '방송일시'

| | 노출(분) | 마더코드 | 상품코드 | 상품명 | 상품군 | 판매단가 | 취급액 | 시 | 분 | 월 | 일 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 20 | 100346 | 201072 | 테이트 남성 셀린니트3종 | 의류 | 39900 | 2099000 | 6 | 0 | 1 | 1 |
| 1 | 20 | 100346 | 201079 | 테이트 여성 셀린니트3종 | 의류 | 39900 | 4371000 | 6 | 0 | 1 | 1 |
| 2 | 20 | 100346 | 201072 | 테이트 남성 셀린니트3종 | 의류 | 39900 | 3262000 | 6 | 20 | 1 | 1 |
| 3 | 20 | 100346 | 201079 | 테이트 여성 셀린니트3종 | 의류 | 39900 | 6955000 | 6 | 20 | 1 | 1 |
| 4 | 20 | 100346 | 201072 | 테이트 남성 셀린니트3종 | 의류 | 39900 | 6672000 | 6 | 40 | 1 | 1 |

시, 분, 월, 일로 나눠서 각각 Column 만들어줌

## 컬럼 별 정리

### '상품군'

```
df_dummy = pd.get_dummies(df_raw['상품군'])
df_dummy = df_raw.join(df_dummy.add_prefix('상품군_'))
```

```
df_dummy = df_dummy.drop(['상품군'], axis=1)
df_dummy.head()
```

| | 노출(분) | 마더코드 | 상품코드 | 상품명 | 판매단가 | 취급액 | 시 | 분 | 월 | 일 | ... | 상품군_가전 | 상품군_건강기능 | 상품군_농수축 | 상품군_생활용품 | 상품군_속옷 | 상품군_의류 | 상품군_이미용 | 상품군_잡화 | 상품군_주방 | 상품군_침구 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 20 | 100346 | 201072 | 테이트 남성 셀린니트3종 | 39900 | 2099000 | 6 | 0 | 1 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 20 | 100346 | 201079 | 테이트 여성 셀린니트3종 | 39900 | 4371000 | 6 | 0 | 1 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 2 | 20 | 100346 | 201072 | 테이트 남성 셀린니트3종 | 39900 | 3262000 | 6 | 20 | 1 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 3 | 20 | 100346 | 201079 | 테이트 여성 셀린니트3종 | 39900 | 6955000 | 6 | 20 | 1 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 4 | 20 | 100346 | 201072 | 테이트 남성 셀린니트3종 | 39900 | 6672000 | 6 | 40 | 1 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

상품군 별로 더미 변수 생성

## 컬럼 별 정리

**'상품명'**

### 성별

```
# 상품명 성별 포함 고려해 새로운 column 추가하기
condition_list=[
    (df_raw['상품명'].str.contains('여성|브라|오모떼|보정팬티|뷰티')),
    (df_raw['상품명'].str.contains('남성'))
]
value_list=['여', '남']
df_raw['성별']=np.select(condition_list,value_list)
df_raw.head()
```

성별을 나타내는 단어들로 성별 열을 추가

```
df_dummy = pd.get_dummies(df_raw['성별'])
df_dummy = df_raw.join(df_dummy.add_prefix('성별_'))

df_dummy = df_dummy.drop(['성별'], axis=1)

df_raw = df_dummy
df_raw.head()
```

더미 변수 생성

| 성별_0 | 성별_남 | 성별_여 |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |

### 무이자/일시불

```
condition_list=[
    (df_raw['상품명'].str.contains('무이자')),
    (df_raw['상품명'].str.contains('일시불'))
]
value_list=['무이자', '일시불']
df_raw['무이자or일시불']=np.select(condition_list,value_list)
df_raw.head()
```

무이자 or 일시불 단어 포함한 열 구분

```
df_dummy = pd.get_dummies(df_raw['무이자or일시불'])
df_dummy = df_raw.join(df_dummy.add_prefix('무이자or일시불='))

df_dummy = df_dummy.drop(['무이자or일시불'], axis=1)

df_raw = df_dummy
df_raw.head()
```

더미 변수 생성

| 무이자or일시불=0 | 무이자or일시불=무이자 | 무이자or일시불=일시불 |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 1 | 0 | 0 |

## 컬럼 별 정리

**‘상품명’**

### 계절

```
condition_list=[
    (df_raw['상품명'].str.contains('여름')),
    (df_raw['상품명'].str.contains('겨울|니트|기모|온수|단열|패딩|무스탕|방한|덕다운|히터|스웨터|롱패딩'))
]
value_list=['여름', '겨울']
df_raw['계절']=np.select(condition_list,value_list)
df_raw.head()
```

계절을 나타내는 단어들로 계절 열을 추가

```
df_dummy = pd.get_dummies(df_raw['계절'])
df_dummy = df_raw.join(df_dummy.add_prefix('계절_'))

df_dummy = df_dummy.drop(['계절'], axis=1)

df_raw = df_dummy
df_raw.head()
```

더미 변수 생성

| 계절_0 | 계절_겨울 | 계절_여름 |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 1 | 0 |

## 컬럼 별 정리

‘컬럼 결과’

| | 노출(분) | 마더코드 | 상품코드 | 판매단가 | 취급액 | 시 | 분 | 월 | 일 | 상품군_가구 | 상품군_가전 | 상품군_건강기능 | 상품군_농수축 | 상품군_생활용품 | 상품군_속옷 | 상품군_의류 | 상품군_이미용 | 상품군_잡화 | 상품군_주방 | 상품군_침구 | 성별_0 | 성별_남 | 성별_여 | 무이자or일시불=무이자 | 무이자or일시불=일시불 | 계절_0 | 계절_겨울 | 계절_여름 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 20 | 100346 | 201072 | 39900 | 2099000 | 6 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **1** | 20 | 100346 | 201079 | 39900 | 4371000 | 6 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| **2** | 20 | 100346 | 201072 | 39900 | 3262000 | 6 | 20 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **3** | 20 | 100346 | 201079 | 39900 | 6955000 | 6 | 20 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| **4** | 20 | 100346 | 201072 | 39900 | 6672000 | 6 | 40 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

## scale 변환 : 평균, 표준 편차 기준

```
df_scale_std=scale(df_x)
df_scale_std=pd.DataFrame(df_scale_std, columns=df_x.columns)
df_scale_std.head()
```

| | 노출(분) | 마더코드 | 상품코드 | 판매단가 | 시 | 분 | 월 | 일 | 상품군_가구 | 상품군_가전 | 상품군_건강기능 | 상품군_농수축 | 상품군_생활용품 | 상품군_속옷 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | -0.10915 | -0.193142 | -0.211097 | -0.55549 | -1.234828 | -1.245633 | -1.617393 | -1.617393 | -0.263809 | -0.413364 | -0.150736 | -0.351171 | -0.291398 | -0.35249 | 2 |
| **1** | -0.10915 | -0.193142 | -0.201628 | -0.55549 | -1.234828 | -1.245633 | -1.617393 | -1.617393 | -0.263809 | -0.413364 | -0.150736 | -0.351171 | -0.291398 | -0.35249 | 2 |
| **2** | -0.10915 | -0.193142 | -0.211097 | -0.55549 | -1.234828 | -0.025233 | -1.617393 | -1.617393 | -0.263809 | -0.413364 | -0.150736 | -0.351171 | -0.291398 | -0.35249 | 2 |
| **3** | -0.10915 | -0.193142 | -0.201628 | -0.55549 | -1.234828 | -0.025233 | -1.617393 | -1.617393 | -0.263809 | -0.413364 | -0.150736 | -0.351171 | -0.291398 | -0.35249 | 2 |
| **4** | -0.10915 | -0.193142 | -0.211097 | -0.55549 | -1.234828 | 1.195167 | -1.617393 | -1.617393 | -0.263809 | -0.413364 | -0.150736 | -0.351171 | -0.291398 | -0.35249 | 2 |

평균, 표준 편차 기준으로 scale 변환 후 아래와 같이 요약통계량으로 scale 변환 결과 확인

| | 노출(분) | 마더코드 | 상품코드 | 판매단가 | 시 | 분 | 월 | 일 | 상품군_가구 | 상품군_가전 | ... | 상품군_침구 | 성별_0 | 성별 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 35379.000 | 35379.000 | 35379.000 | 35379.000 | 35379.000 | 35379.000 | 35379.000 | 35379.000 | 35379.000 | 35379.000 | ... | 35379.000 | 35379.000 | 35379 |
| **mean** | -0.000 | 0.000 | 0.000 | -0.000 | -0.000 | -0.000 | -0.000 | -0.000 | -0.000 | -0.000 | ... | 0.000 | 0.000 | -0 |
| **std** | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | ... | 1.000 | 1.000 | 1 |
| **min** | -5.935 | -1.566 | -1.661 | -0.599 | -2.148 | -1.246 | -1.617 | -1.617 | -0.264 | -0.413 | ... | -0.138 | -2.324 | -0 |
| **25%** | -0.109 | -0.951 | -0.891 | -0.525 | -0.778 | -1.246 | -0.734 | -0.734 | -0.264 | -0.413 | ... | -0.138 | 0.430 | -0 |
| **50%** | -0.109 | -0.165 | -0.072 | -0.461 | 0.135 | -0.025 | -0.145 | -0.145 | -0.264 | -0.413 | ... | -0.138 | 0.430 | -0 |
| **75%** | -0.109 | 0.831 | 0.881 | 0.018 | 0.897 | 1.195 | 0.738 | 0.738 | -0.264 | -0.413 | ... | -0.138 | 0.430 | -0 |
| **max** | 6.364 | 1.803 | 1.738 | 12.040 | 1.353 | 1.805 | 1.622 | 1.622 | 3.791 | 2.419 | ... | 7.231 | 0.430 | 3 |

# 02 모델 생성

## 데이터 구성하기

```
df_train_x,df_test_x, df_train_y, df_test_y = train_test_split(df_x, df_y,test_size = 0.05/2, shuffle = True, random_state = 1234)

print('train data X size : {}'.format(df_train_x.shape))
print('train data Y size : {}'.format(df_train_y.shape))
print('test data X size : {}'.format(df_test_x.shape))
print('test data X size : {}'.format(df_test_y.shape))
```

```
train data X size : (34494, 28)
train data Y size : (34494,)
test data X size : (885, 28)
test data X size : (885,)
```

```
df_train_x, df_val_x, df_train_y, df_val_y = train_test_split(df_train_x, df_train_y ,test_size = 0.05/1.95, shuffle = True, random_st

print('train data X size : {}'.format(df_train_x.shape))
print('train data Y size : {}'.format(df_train_y.shape))
print('validation data X size : {}'.format(df_val_x.shape))
print('validation data X size : {}'.format(df_val_y.shape))
```

```
train data X size : (33609, 28)
train data Y size : (33609,)
validation data X size : (885, 28)
validation data X size : (885,)
```

```
print('train data X size : {}'.format(df_train_x.shape))
print('train data Y size : {}'.format(df_train_y.shape))
print('validation data X size : {}'.format(df_val_x.shape))
print('validation data X size : {}'.format(df_val_y.shape))
print('test data X size : {}'.format(df_test_x.shape))
print('test data X size : {}'.format(df_test_y.shape))
```

```
train data X size : (33609, 28)
train data Y size : (33609,)
validation data X size : (885, 28)
validation data X size : (885,)
test data X size : (885, 28)
test data X size : (885,)
```

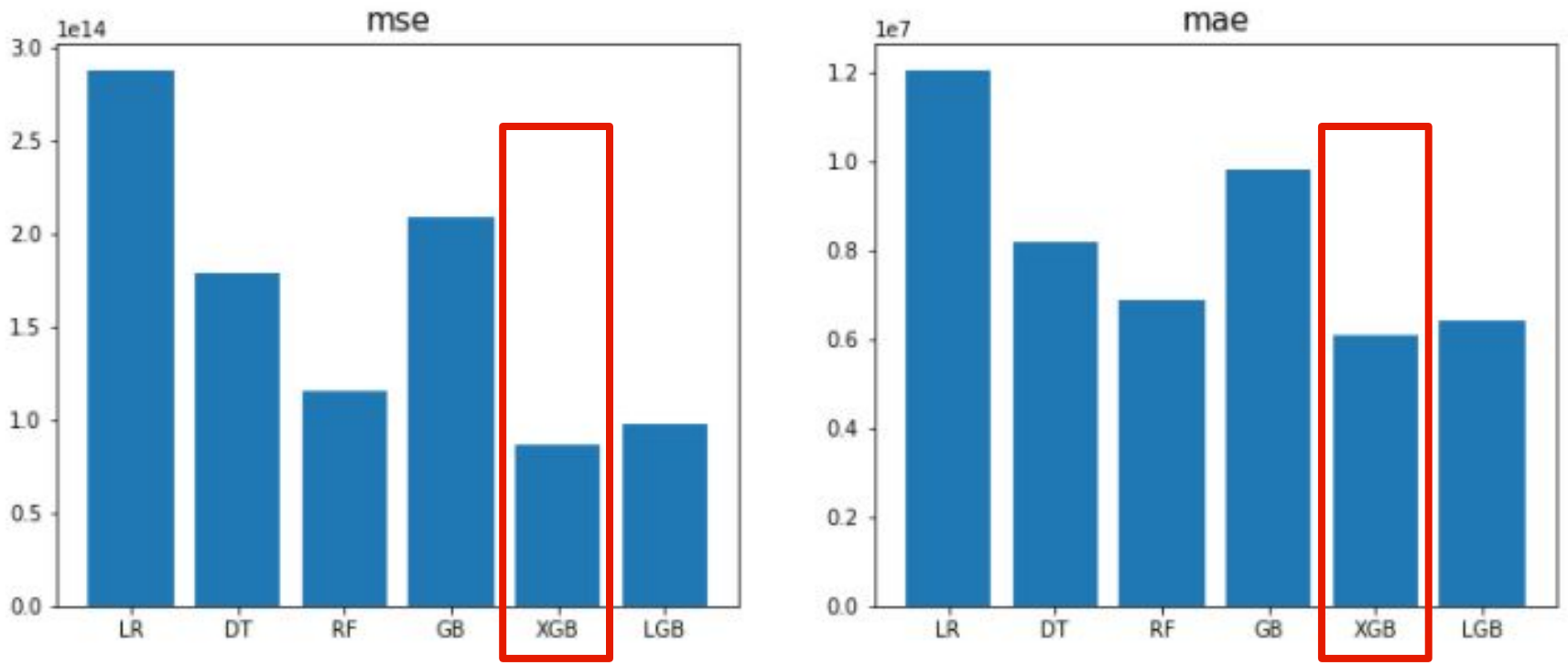
## 모델 생성 및 결과

| | training score | validation score | test score |
|---|---|---|---|
| LinearRegression | 0.313 | 0.328 | 0.356 |
| DecisionTreeRegressor | 0.940 | 0.456 | 0.599 |
| RandomForestRegressor | 0.915 | 0.673 | 0.742 |
| GradientBoostingRegressor | 0.499 | 0.509 | 0.531 |
| **xgboost** | 0.862 | 0.768 | **0.806** |
| lightgbm | 0.844 | 0.759 | 0.781 |

**>> xgboost 모델에서 80.6% 정확도를 보임**

# 03 모델 평가

## mse, mae로 모델 평가




>> mse, mae 결과 둘 다 **XGBoost** 가 가장 낮게 나온 것을 확인 할 수 있음

# 04 결과

## test 데이터에 적용

**취급액 예측 결과**

```
pred_df_s=pd.DataFrame(data={'취급액':s_result.tolist()})
pred_df_s
```

| | 취급액 |
|---|---|
| **0** | -16814724.00 |
| **1** | -16814724.00 |
| **2** | -14753872.00 |
| **3** | 3016597.25 |
| **4** | 3016597.25 |
| **...** | ... |
| **2711** | 18088802.00 |
| **2712** | 16102098.00 |
| **2713** | 23272366.00 |
| **2714** | 14965814.00 |
| **2715** | 15845472.00 |

2716 rows × 1 columns

# 2. 최적 수익을 고려한 편성 최적화 방안(모형) 제시

# 01 시청률 데이터 활용

## 시청률 데이터 가공

시청률 데이터 원본

| 날짜 | 시간 | 시청률 |
|---|---|---|
| 1월1일 | 2:00 | 0.0007 |
| 1월1일 | 2:20 | 0 |
| 1월1일 | 2:40 | 0 |
| 1월1일 | 3:00 | 0 |
| 1월1일 | 3:20 | 0 |
| 1월1일 | 3:40 | 0.01115 |
| 1월1일 | 4:00 | 0.011 |
| 1월1일 | 4:20 | 0.01115 |
| 1월1일 | 4:40 | 0.011 |
| 1월1일 | 5:00 | 0.011 |
| 1월1일 | 5:20 | 0.011 |
| 1월1일 | 5:40 | 0.00055 |
| 1월1일 | 6:00 | 0 |
| 1월1일 | 6:20 | 0 |

시청률_평균.csv

날짜별, 시간별(20분 단위)로
시청률 평균 내어 데이터 가공

시청률 데이터 전처리
.ipynb

| 시청률 | 월 | 일 | 시 | 분 |
|---|---|---|---|---|
| 0.0007 | 1 | 1 | 2 | 0 |
| 0 | 1 | 1 | 2 | 20 |
| 0 | 1 | 1 | 2 | 40 |
| 0 | 1 | 1 | 3 | 0 |
| 0 | 1 | 1 | 3 | 20 |
| 0.01115 | 1 | 1 | 3 | 40 |
| 0.011 | 1 | 1 | 4 | 0 |
| 0.01115 | 1 | 1 | 4 | 20 |
| 0.011 | 1 | 1 | 4 | 40 |
| 0.011 | 1 | 1 | 5 | 0 |
| 0.011 | 1 | 1 | 5 | 20 |
| 0.00055 | 1 | 1 | 5 | 40 |

df_time_mean.csv

날짜와 시간을 월, 일, 시, 분으로 나눠서
실적 데이터와 합칠 수 있도록 가공

# 02 요일/시간대별 분석

## 요일 컬럼 생성

```
df_raw['요일']= pd.DataFrame(a)
df_raw
```

| | 방송일시 | 노출(분) | 마더코드 | 상품코드 | 상품명 | 상품군 | 판매단가 | 취급액 | 요일 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2019.1.1 6:00 | 20.0 | 100346 | 201072 | 테이트 남성 셀린니트3종 | 의류 | 39900 | 2099000.0 | 1 |
| 1 | 2019.1.1 6:00 | NaN | 100346 | 201079 | 테이트 여성 셀린니트3종 | 의류 | 39900 | 4371000.0 | 1 |
| 2 | 2019.1.1 6:20 | 20.0 | 100346 | 201072 | 테이트 남성 셀린니트3종 | 의류 | 39900 | 3262000.0 | 1 |
| 3 | 2019.1.1 6:20 | NaN | 100346 | 201079 | 테이트 여성 셀린니트3종 | 의류 | 39900 | 6955000.0 | 1 |
| 4 | 2019.1.1 6:40 | 20.0 | 100346 | 201072 | 테이트 남성 셀린니트3종 | 의류 | 39900 | 6672000.0 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 38304 | 2020.1.1 0:20 | 20.0 | 100073 | 200196 | 삼성화재 행복한파트너 주택화재보험(1912) | 무형 | 0 | NaN | 2 |
| 38305 | 2020.1.1 0:40 | 20.0 | 100073 | 200196 | 삼성화재 행복한파트너 주택화재보험(1912) | 무형 | 0 | NaN | 2 |
| 38306 | 2020.1.1 1:00 | 20.0 | 100073 | 200196 | 삼성화재 행복한파트너 주택화재보험(1912) | 무형 | 0 | NaN | 2 |
| 38307 | 2020.1.1 1:20 | 20.0 | 100490 | 201478 | 더케이 예다함 상조서비스(티포트) | 무형 | 0 | NaN | 2 |
| 38308 | 2020.1.1 1:40 | 17.0 | 100490 | 201478 | 더케이 예다함 상조서비스(티포트) | 무형 | 0 | NaN | 2 |

38309 rows × 9 columns

**>> “방송일시” 컬럼 값을 기준으로 datetime.dayofweek 을 이용하여 “요일” 컬럼 생성**

# 02 요일/시간대별 분석

## 요일별 최대 시청률과 최다 주문량 수 조회

예시

```
pd.set_option('display.max_columns', 100)
tues_df = merge_df[merge_df['요일']==1]
tues_df['시청률'].sort_values(ascending=False)
```

```
30195    0.0411
31472    0.0411
30196    0.0411
30828    0.0411
31471    0.0411
```

< 최대 시청률 조회 >

```
tues_df[tues_df['시청률']==0.0411]['주문량'].sort_values(ascending=False)
```

```
30828    1271
32075     590
31472     298
31471     167
30195     117
30196      83
Name: 주문량, dtype: int32
```

< 최다 주문량 조회 >

**>> ex) 화요일의 최대 시청률= 0.0411, 시청률이 가장 높을때의 주문량 수 = 1271**

# 02 요일/시간대별 분석

## 요일별 최고 시청률 최다 주문량 상품군 조회 결과

```
thurs_df[(thurs_df['주문량']==430) & (thurs_df['시청률']==0.0411)]
```

| | 노출(분) | 마더코드 | 상품코드 | 판매단가 | 취급액 | 요일 | 시 | 분 | 월 | 일 | 상품군_가구 | 상품군_가전 | 상품군_건강기능 | 상품군_농수축 | 상품군_생활용품 | 상품군_속옷 | 상품군_의류 | 상품군_이미용 | 상품군_잡화 | 상품군_주방 | 상품군_침구 | 성별_0 | 성별_남 | 성별_여 | 무이자or일시불=0 | 무이자or일시불=무이자 | 무이자or일시불=일시불 | 계절_0 | 계절_겨울 | 계절_여름 | 주문량 | 시청률 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| !248 | 20 | 100062 | 200152 | 149000 | 64092000 | 3 | 19 | 20 | 11 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 430 | 0.0411 |

>> 상품의 상품군, 방송시간대, 상품코드 등의 정보를 참고하여 추후 방송편성에 활용

# 03 상품군별 데이터 분석

## 상품군에 따라 데이터 분리, 세부 분석

**상품군 일람**

- 0: 주방
- 1: 가전
- 2: 의류
- 3: 속옷
- 4: 잡화
- 5: 농수축
- 6: 생활용품
- 7: 가구
- 8: 이미용
- 9: 건강기능
- 10: 침구

**>> sys.module과 setattr로 상품군별 데이터 분리**

```
# 상품군 기준으로 데이터 분리하기
mod=sys.modules[__name__]

for i in range(11):
    setattr(mod, 'df_{}'.format(i), load_df[load_df['상품군']==i])

print('데이터 분리 완료')
```

```
1  # 데이터 확인
2  df_2['상품군'].value_counts()
```

```
2    4340
Name: 상품군, dtype: int64
```

**>> 각 상품군별로 인기 상품을 분석하기 위한 준비!**

# 03 상품군별 데이터 분석

## 인기 상품 분류 기준 : 주문량과 취급액

### 주문량

- 절대적인 주문량 기준
- **시청자들의 선호도**와 깊은 연관성
- 시청자가 부담없어하는 가격대를 파악 가능

### 취급액

- 절대 판매액이 기준
- 기본 가격대가 높은 상품들이 상위에 포진되기 쉽다는 한계
- 주문량 기준보다 **상품별 매출 최적 시간대** 탐색 용이

- **분석 방법**

**>> 취급액과 주문량을 기준으로 상위 100개 행 추출**
**>> '취급액 금액 상위 100', '주문량 횟수 상위 100' 의 교집합 추출**
**>> 각 상품별 데이터 추출, 최적 시간대 탐색**

# 03 상품군별 데이터 분석

## 주문량 기준 TOP 100

예시

```
# 주문량 기준
df_0 = add_column(df_0)
df_0 = df_0.sort_values(by=['주문량'], ascending=False)
df_0
```

>> 주문량 기준으로 내림차순 정렬

```
# 인기상품 top 100(주문량 기준)
top_100_00 = df_0[:100]
top_100_00
```

| | 방송일시 | 노출(분) | 마더코드 | 상품코드 | 상품명 | 상품군 | 판매단가 | 취급액 | 성별 포함 | 년 | 월 | 일자 | 시 | 분 | 초 | 주문량 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **38086** | 2019-12-29 16:00:00 | 20.0 | 100724 | 202115 | IH 옛 가마솥 세트 | 0 | 62000 | 167261000 | 0 | 2019 | 12 | 29 | 16 | 0 | 0 | 2697 |
| **34907** | 2019-11-28 16:20:00 | 20.0 | 100255 | 200869 | 한일 대용량 스텐 분쇄믹서기 | 0 | 109800 | 225084000 | 0 | 2019 | 11 | 28 | 16 | 20 | 0 | 2049 |
| **38059** | 2019-12-29 08:00:00 | 20.0 | 100236 | 200806 | 벨라홈 스마트 멀티포트 1+1 세트 | 0 | 39800 | 68162000 | 0 | 2019 | 12 | 29 | 8 | 0 | 0 | 1712 |
| **23** | 2019-01-01 12:40:00 | 20.0 | 100088 | 200236 | 에코라믹 통주물 스톤 냄비세트 | 0 | 60900 | 99235000 | 0 | 2019 | 1 | 1 | 12 | 40 | 0 | 1629 |
| **37962** | 2019-12-28 09:00:00 | 20.0 | 100264 | 200878 | 셀렉프로 뚝배기 전기밥솥(멀티쿠커) | 0 | 79000 | 128382000 | 0 | 2019 | 12 | 28 | 9 | 0 | 0 | 1625 |
| **14642** | 2019-05-18 09:40:00 | 20.0 | 100478 | 201453 | 파뷔에 항균도마 1+1세트 | 0 | 49800 | 79405000 | 0 | 2019 | 5 | 18 | 9 | 40 | 0 | 1594 |

>> TOP 100 상품 추출

# 03 상품군별 데이터 분석

## 취급액 기준 TOP 100(가전)

예시

```
#'주방'
df_0_sold = df_0.sort_values(by=['취급액'], ascending=False)
```

>> '취급량' 기준으로 내림차순 정렬

```
# 인기상품 top 100(취급액 기준)
df_0_sold = df_0_sold.reset_index(drop=True)
top_100_0 = df_0_sold.loc[:99]
top_100_0
```

| | 방송일시 | 노출(분) | 마더코드 | 상품코드 | 상품명 | 상품군 | 판매단가 | 취급액 | 성별 포함 | 년 | 월 | 일자 | 시 | 분 | 초 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2019-11-28 16:20:00 | 20.0 | 100255 | 200869 | 한일 대용량 스텐 분쇄믹서기 | 0 | 109800 | 225084000 | 0 | 2019 | 11 | 28 | 16 | 20 | 0 |
| 1 | 2019-12-29 16:00:00 | 20.0 | 100724 | 202115 | IH 옛 가마솥 세트 | 0 | 62000 | 167261000 | 0 | 2019 | 12 | 29 | 16 | 0 | 0 |
| 2 | 2019-11-17 11:00:00 | 20.0 | 100255 | 200869 | 한일 대용량 스텐 분쇄믹서기 | 0 | 109800 | 142764000 | 0 | 2019 | 11 | 17 | 11 | 0 | 0 |
| 3 | 2019-12-29 13:00:00 | 20.0 | 100638 | 201956 | 램프쿡 자동회전냄비 | 0 | 109000 | 141159000 | 0 | 2019 | 12 | 29 | 13 | 0 | 0 |
| 4 | 2019-12-28 09:00:00 | 20.0 | 100264 | 200878 | 셀렉프로 뚝배기 전기밥솥(멀티쿠커) | 0 | 79000 | 128382000 | 0 | 2019 | 12 | 28 | 9 | 0 | 0 |
| 5 | 2019-11-24 17:00:00 | 20.0 | 100837 | 202467 | 무이자 쿠쿠전기밥솥 10인용(CRP-QS107FG/FS) | 0 | 218000 | 119573000 | 0 | 2019 | 11 | 24 | 17 | 0 | 0 |

>> TOP 100 상품 추출

# 03 상품군별 데이터 분석

## 주문량 TOP 100 & 취급액 TOP 100 교집합 추출

**예시**

**중복값 제거**

```
# 상품명별, 마더코드별 중복 제거
# 상품명
name_li_00 = list(set(top_100_00['상품명']))
# print(len(name_li_00))
name_li_00 # 상품명 100개 중 88개가 중복
```

```
# 각 상품명별, 마더코드별 중복 삭제
# 상품명
name_li_0 = list(set(top_100_0['상품명']))
# print(len(name_li_0))
name_li_0 # 상품명 100개 중 75개가 겹침
```

>> 같은 상품명이어도 광고하는 시간대가 다르면 다른 행으로 구분

**교집합 추출**

```
# 두 기준 모두에 포함되는 상품명
popular_name_0 = list(set(name_li_0) & set(name_li_00))
popular_name_0
```

```
['셀렉프로 뚝배기 전기밥솥(멀티쿠커)',
 '파뷔에 항균도마 1+1+칼세트',
 '한일 대용량 스텐 분쇄믹서기',
 '초특가  파뷔에 팬지 대형 후라이팬 세트',
 '에코라믹 통주물 스톤 냄비세트',
 '램프쿡 자동회전냄비',
 'IH 옛 가마솥 세트',
 '파뷔에 항균도마 1+1세트',
 '에코라믹 컬링스톤프라이팬세트']
```

# 03 상품군별 데이터 분석

## 각 상품별 데이터 추출, 최적 시간대 탐색

예시

### 1 교집합에 포함된 상품 데이터프레임 생성

```
filter_df1 = filter_dataframe_for_list(load_df, '상품명', popular_name_1)
filter_df1.head(10)
```

| | 방송일시 | 노출(분) | 마더코드 | 상품코드 | 상품명 | 상품군 | 판매단가 | 취급액 |
|---|---|---|---|---|---|---|---|---|
| 32425 | 2019-11-03 09:20:00 | 30.0 | 100091 | 200248 | (1등급)221L_딤채김치냉장고 | 1 | 899000 | 5028000 |
| 32427 | 2019-11-03 09:50:00 | 30.0 | 100091 | 200248 | (1등급)221L_딤채김치냉장고 | 1 | 899000 | 15134000 |
| 32919 | 2019-11-07 22:20:00 | 30.0 | 100091 | 200248 | (1등급)221L_딤채김치냉장고 | 1 | 899000 | 2613000 |
| 32921 | 2019-11-07 22:50:00 | 30.0 | 100091 | 200248 | (1등급)221L_딤채김치냉장고 | 1 | 899000 | 32642000 |
| 33087 | 2019-11-09 13:20:00 | 20.0 | 100091 | 200248 | (1등급)221L_딤채김치냉장고 | 1 | 899000 | 7445000 |

### 2 모든 상품군의 교집합 데이터프레임 병합

```
# 각 인기 상품 데이터프레임 병합(효율성 고려)
filter_li = [filter_df0, filter_df1, filter_df2, filter_df3,
             filter_df4, filter_df5, filter_df6, filter_df7,
             filter_df8, filter_df9, filter_df10]
popular_product_df = pd.concat(filter_li)
popular_product_df.head(10)
```

>> 코드 반복 최소화 목적

### 3 각 상품별로 데이터프레임 분리

```
# 상품명 기준으로 데이터 분리
df_li = []
total_popular_name = list(set(popular_product_df['상품명']))
mod=sys.modules[__name__]

for i in range(len(total_popular_name)):
    setattr(mod, 'product_df_{}'.format(i), load_df[load_df['상품명']==total_popular_name[i]][:1]) # 전체 122개
```

>> '취급액' 기준으로 내림차순 정렬
>> 각 상품별 첫 번째 행 추출

# 03 상품군별 데이터 분석

## 각 상품별 데이터 추출, 최적 시간대 탐색

예시

1

```
# 상품별 데이터프레임 모으기
raw_li = []
for i in range(123):
    raw_li.append('product_df_{}'.format(i))
```

2

```
# 데이터 합치기
for i in range(1, 123):
    product_df_0 = dataframe_concat(product_df_0, eval(raw_li[i]))
```

>> 각 상품별 데이터를 하나의 데이터프레임으로 병합

>> 상품별 광고 최적 시간대 추출

3

| | 방송일시 | 노출(분) | 마더코드 | 상품코드 | 상품명 | 상품군 | 판매단가 |
|---|---|---|---|---|---|---|---|
| 16784 | 2019-06-07 08:00:00 | 30.0 | 100478 | 201452 | 파뷔에 항균도마 1+1+칼세트 | 0 | 49800 |
| 21 | 2019-01-01 12:00:00 | 20.0 | 100088 | 200236 | 에코라믹 통주물 스톤 냄비세트 | 0 | 60900 |
| 12634 | 2019-05-02 12:00:00 | 20.0 | 100475 | 201446 | 초특가 파뷔에 팬지 대형 후라이팬 세트 | 0 | 59800 |
| 28406 | 2019-09-25 06:00:00 | 20.0 | 100724 | 202115 | IH 옛 가마솥 세트 | 0 | 62000 |
| 198 | 2019-01-03 14:00:00 | 20.0 | 100255 | 200868 | 한일 대용량 스텐 분쇄믹서기 | 0 | 109800 |
| 37833 | 2019-12-26 19:20:00 | 20.0 | 100638 | 201956 | 램프쿡 자동회전냄비 | 0 | 109000 |
| 9609 | 2019-04-06 07:00:00 | 20.0 | 100478 | 201453 | 파뷔에 항균도마 1+1세트 | 0 | 49800 |
| 37228 | 2019-12-20 08:20:00 | 20.0 | 100264 | 200878 | 셀렉프로 뚝배기 전기밥솥(멀티쿠커) | 0 | 79000 |
| 45 | 2019-01-01 20:00:00 | 20.0 | 100150 | 200533 | 일시불 LG 통돌이 세탁기 | 1 | 439000 |
| 35040 | 2019-11-29 22:20:00 | 20.0 | 100148 | 200412 | LG 울트라HD TV 65UM7900BNA | 1 | 1700000 |

# 04 편성표 최적화

## 편성 최적화 기준 설정

**Main**

A. 요일별 가장 시청률이 높았던 상품군
B. 각 인기상품별로 추출한 최적화 시간대

**Sub**

C. 9~10월 중 상품군 분포 추이

# 04 결론 : 편성표 최적화

## 편성표 예시(금요일 기준)

| | 방송일시 | 노출(분) | 마더코드 | 상품코드 | 상품명 | 상품군 | 판매단가 |
|---|---|---|---|---|---|---|---|
| 0 | 6:00:00 | 20 | 100416 | 201315 | 코몽트 남성 메쉬티셔츠 7종 | 2 | 39900 |
| 1 | 6:20:00 | 20 | 100330 | 201036 | LG생활건강 샤프란아우라 고농축 섬유유연제 | 6 | 30900 |
| 2 | 6:40:00 | 20 | 100478 | 201453 | 파뷔에 항균도마 1+1세트 | 0 | |
| 3 | 7:00:00 | 20 | 100452 | 201395 | NNF SS트레이닝 세트 | 2 | |
| 4 | 7:20:00 | 20 | 100226 | 201019 | 코이모 리빙박스 8종 | 6 | |
| 5 | 7:40:00 | 20 | 100069 | 200189 | 칼리베이직 2019 패브릭 백 4종 (시즌 3) | 4 | |
| 6 | 8:00:00 | 20 | 100248 | 200830 | [K-SWISS] 19 F/W 남성 패딩 트랙수트 2종 | 2 | 79900 |
| 7 | 8:20:00 | 20 | 100354 | 201135 | 마르엘라로사티 에코무스탕1종 | 2 | 99000 |
| 8 | 8:40:00 | 20 | 100635 | 201945 | 바로바로 무선청소기 | 6 | 89000 |
| 9 | 9:00:00 | 20 | 100416 | 201317 | 코몽트 남성 이너티셔츠 7종 | 2 | 39900 |
| 10 | 9:20:00 | 20 | 100753 | 202202 | 아가타 골든 마스터 2종 (펌프스1종+플랫슈즈1종) | 4 | 59900 |
| 11 | 9:40:00 | 20 | 100444 | 201369 | 쿠미투니카 코르셋 하이웨스트 팬티세트 | 3 | 59900 |
| 12 | 10:00:00 | 20 | 100832 | 202447 | 청정수산 완도활전복 中사이즈 26미 | 5 | 59900 |
| | | 20 | 100452 | 201401 | NNF 소프트 베스트 앙상블 | 2 | 29900 |
| | | 20 | 100139 | 200392 | 보몽드 엘사 자수 쿨 시어서커 침구세트 SK(슈퍼킹) | 10 | 89900 |
| | | 20 | 100323 | 201488 | 피시원 국내산 햇 손질문어 7팩 | 5 | 49900 |

1

- **일일 상품군 비중**
  - 요일별/시간대별 최고 시청률 상품군 비중 우위

2

- **상품-시간대 매칭**
  각 인기상품별 매출 최적 시간대 기준으로 매칭

**>> 1 과 2 모두를 충족하는 상품 배치를 최우선으로 함**

# 감사합니다.