MODU PARKING APP ANALYSIS

어플리케이션 이용자 수 예측과 전반적인 운영전략에 대하여

안태윤

CONTENT

01 프로젝트 개요

- 모두의 주차장 어플리케이션 이용자 데이터를 분석해 이용자별 이용 건수를 예측
- 이용건수 예측분석은 이용자별 일별 이용건수 예측과 이용자별 월 합계 이용건수 예측으로 나누어서 분석
- 모두의 주차장 어플리케이션 이용자 데이터를 분석해 서비스 개선 방향, 운영전략등을 수립
- 분석데이터 : 실전db.csv(모두컴퍼니 제공 모두의주차장 어플리케이션 이용자 데이터), 서울시_기상데이터

02 데이터 전처리

```
df_ori = pd.read_csv('실전db.csv', encoding='cp949')
df_ori.head()
```

	USER_ID	JOIN_DATE	D_TYPE	STORE_ID	GOODS_TYPE	DATE	COUNT	AD1
0	2858	2014-01-07	AA	1892	A	2020-01-01	1	GN
1	5647	2014-02-14	BB	182009	A	2020-01-01	1	J
2	33314	2014-11-20	B					
3	37001	2014-12-04	B					
4	37819	2014-12-07	A					

```
#원본 데이터 변경을 막기위해 데이터프레임 변경
df=df_ori.copy()
df
```

- 모두 컴퍼니제공 모두의 주차장 어플 이용자 데이터(실전db.csv) 로딩
- 이후 원본데이터 변경을 막기위해 데이터프레임 복사본 생성

02 데이터 전처리

```
#데이터 정보 확인
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 879271 entries, 0 to 879270
Data columns (total 8 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   USER_ID     879271 non-null  int64
 1   JOIN_DATE   879271 non-null  object
 2   D_TYPE      879271 non-null  object
 3   STORE_ID    879271 non-null  int64
 4   GOODS_TYPE  879271 non-null  object
 5   DATE        879271 non-null  object
 6   COUNT       879271 non-null  int64
 7   AD1         879271 non-null  object
dtypes: int64(3), object(5)
memory usage: 53.7+ MB
None
```

- 데이터 정보 확인
- 8개 컬럼으로 구성
- 정수형과 문자형으로 구성된 범주형 컬럼들

02 데이터 전처리

- 이상치의 경우 프로젝트의 목표(정상적인 이용자의 한달 사용 건수 예측)를 고려해일반적인 이용자로 보이지 않는 데이터를 이상치로 판단
- 다음 조건에 해당하는 이용자는 이상치로 처리
  1. 연속적으로 이용
  2. 데이터가 5개 이상
  3. AD10이 계속해서 바뀜
  4. COUNT 값의 평균이 비정상적으로 높음

```
#USER_ID=999665 인 이용자에 대한 처리
df=df[df['USER_ID']!=999665]
```

조건에 부합하는
이상치 제거

03 EDA & FEATURE ENGINEERING

```
#USER_ID별 COUNT의 MEAN값
df_uc=df[['USER_ID','COUNT']]
df_ucmean=df_uc.groupby('USER_ID').mean().reset_index()
df_ucmean.sort_values(by='COUNT',ascending=False)
```

	USER_ID	COUNT
130780	1635143	23.00
108481	1544719	19.00
157735	1775410	17.00
121597	1599008	15.00
8713	430741	11.00
...	...	...
56886	1160108	1.00
56887	1160109	1.00
56888	1160114	1.00
56889	1160128	1.00
165423	1830598	1.00

165424 rows × 2 columns

- **USER_ID와 목표변수(COUNT)와의 상관관계 확인**
- **USER_ID 별로 COUNT의 평균값의 차이가 존재**
- **USER_ID를 특성에 따라 3 그룹으로 나누어 Feature로 구성**

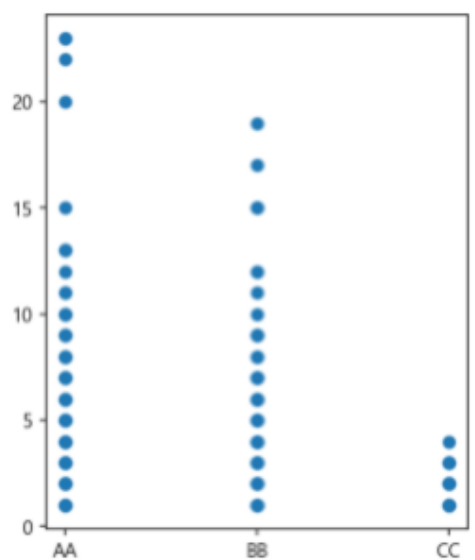
03 EDA & FEATURE ENGINEERING

```
# JOIN_DATE별 COUNT값
plt.figure(figsize=(10,5))
sns.lineplot(df_g.groupby("JOIN_DATE")["COUNT"].sum().keys(),df_g.groupby("JOIN_DATE")["COUNT"].sum())
plt.grid()
plt.title('가입년도별 이용횟수', fontsize=18)
plt.ylabel('이용횟수', fontsize=16)
plt.xlabel('가입년도', fontsize=16)
plt.show()
```

- JOIN_DATE와 목표변수(COUNT)와의 상관관계 확인
- 가입년도가 최근일수록 목표변수(COUNT)의 값이 높아지는 것을 확인
- 가입년도에 따른 Feature 구성

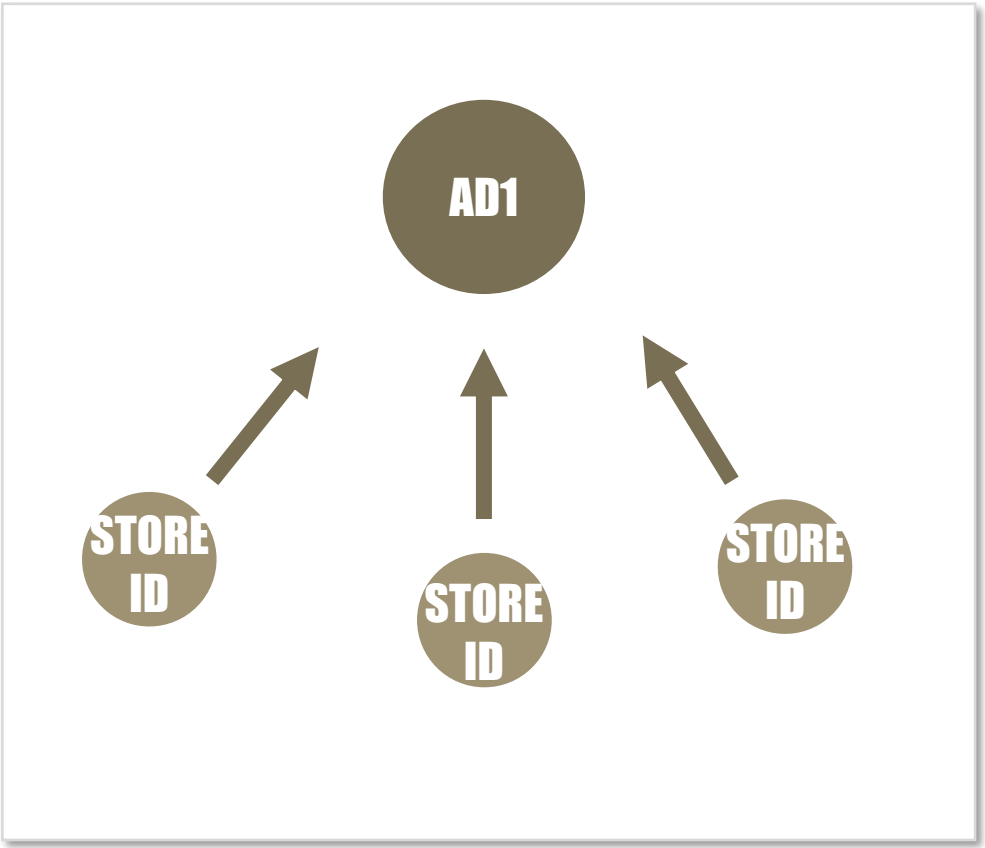
03 EDA & FEATURE ENGINEERING

```
#D_TYPE과 목표변수와의 상관관계
dx=df['D_TYPE']
dy=df['COUNT']
plt.figure(figsize=(4,5))
plt.scatter(dx,dy)
plt.show()
```



- D_TYPE과 목표변수(COUNT)와의 상관관계 확인
- 특정 D_TYPE에서만 높은 COUNT값이 기록되는 등 D_TYPE별로 COUNT값의 분포가 다름

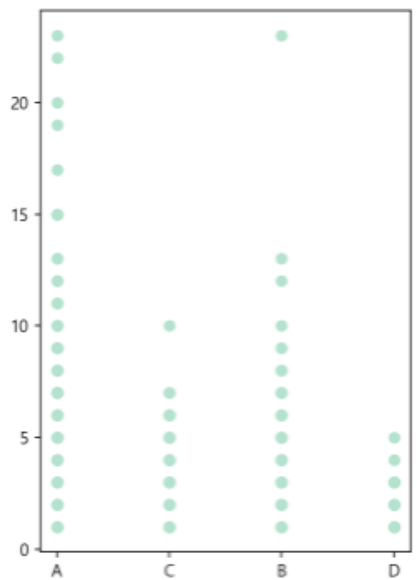
03 EDA & FEATURE ENGINEERING



- 모든 STORE_ID는 AD1에 종속되는 데이터
- AD1이 STORE_ID의 의미를 설명가능

03 EDA & FEATURE ENGINEERING

```
#GOODS_TYPE과 목표변수와의 상관관계
gx=df['GOODS_TYPE']
gy=df['COUNT']
plt.figure(figsize=(4,6))
plt.scatter(gx,gy)
plt.show()
```



- GOODS_TYPE과 목표변수(COUNT)와의 상관관계 확인
- 특정 GOODS_TYPE에서만 높은 COUNT값이 기록되는 등 GOODS_TYPE별로 COUNT값의 분포가 다름

03 EDA & FEATURE ENGINEERING

```
# 공휴일 컬럼 만들기
# 2020년 휴일 리스트
holiday_list = ['2020-01-01', '2020-01-24', '2020-01-25', '2020-01-26', '2020-01-27', '2020-03-01', '2020-05-05', '2020-04-15', '2020-04-30
day_type_list = []
for z in range(len(df_day)) :
    if df_day['DATE'].loc[z].strftime('%Y-%m-%d') in holiday_list :
        day_type_list.append("공휴일")
    elif df_day['DATE'].loc[z].weekday() > 4 :
        day_type_list.append("주말")
    else :
        day_type_list.append("주중")
df_day["DAY_TYPE"] = day_type_list
```

Slide Type Skip

```
# DAY TYPE별 분류
df_day_0= df_day[df_day['DAY_TYPE']=='주말']
df_day_1= df_day[df_day['DAY_TYPE']=='공휴일']
df_day_2= df_day[df_day['DAY_TYPE']=='주중']
```

- DATE컬럼을 활용하기 위해 반복되는 주기성을 포착
- 반복되는 주기성을 가진 주중,주말,공휴일로 DATE컬럼을 분류

03 EDA & FEATURE ENGINEERING

```
#주중과 COUNT의 관계
day_count(df_day_2)
```

```
주중 데이터입니다.

데이터 중 4보다 큰 COUNT 항의 개수 : 167
데이터 중 1보다 큰 COUNT 항의 개수 : 10937

전체 주말데이터의 개수 : 645405
데이터 중 5회 이상 사용한 사람의 비율 : 0.025875225633516942
데이터 중 2회 이상 사용한 사람의 비율 : 1.6945948667890702
```

```
#주말과 COUNT의 관계
day_count(df_day_0)
```

```
주말 데이터입니다.

데이터 중 4보다 큰 COUNT 항의 개수 : 106
데이터 중 1보다 큰 COUNT 항의 개수 : 3721

전체 주말데이터의 개수 : 208282
데이터 중 5회 이상 사용한 사람의 비율 : 0.05089253992183674
데이터 중 2회 이상 사용한 사람의 비율 : 1.7865201985769292
```

- **DATE 타입(주중, 주말, 공휴일)별 목표변수(COUNT)와의 관계를 살펴보니 주중과 주말에서 더 높은 COUNT를 보이는 경향 존재**

03 EDA & FEATURE ENGINEERING

```
seoul=['JR', 'J', 'YO', 'SOD', 'GJ', 'DM', 'JRR', 'SB', 'GB', 'DB', 'NW', 'EP', 'SD', 'MP', 'YC', 'GS', 'GR', 'GHN', 'YD', 'DJ', 'GW', 'SC
in_seoul = []
out_of_seoul = []
for x in df['AD1'].unique() :
    new_df = df[df['AD1']==x]
    if x in seoul :
        in_seoul.append(new_df['STORE_ID'].nunique())
    else :
        out_of_seoul.append(new_df['STORE_ID'].nunique())

print("서울 AD1 STORE_ID 평균 개수 :",sum(in_seoul)/25)
print("서울 외 AD1 STORE_ID 평균 개수 :",sum(out_of_seoul)/60)
```

```
서울 AD1 STORE_ID 평균 개수 : 28.64
서울 외 AD1 STORE_ID 평균 개수 : 5.733333333333333
```

- 이니셜에 근거해 AD1의 행정구역 유추 후 서울지역과 비서울지역의 STORE_ID개수 비교
- 서울지역의 STORE_ID 수가 압도적으로 많아 AD1을 서울지역과 비서울 지역으로 분류한 가설을 채택

03 EDA & FEATURE ENGINEERING

```
sc=0
ns=0
nsc=0
nns=0
ms=0
mns=0
for i in df.index:
        val = df.loc[i, 'AD1']
        if val in seoul :
            sc=sc+df.loc[i,'COUNT']
            if df.loc[i,'COUNT']>ms :
                ms=df.loc[i,'COUNT']
            ns=ns+1
        else :
            nsc=nsc+df.loc[i,'COUNT']
            if df.loc[i,'COUNT']>mns :
                mns=df.loc[i,'COUNT']
            nns=nns+1

print("서울 AD1 COUNT 평균 개수 :",sc/ns,"서울 AD1 COUNT MAX값 :",ms)
print("서울 외 AD1 COUNT 평균 개수 :",nsc/nns,"서울 AD1 COUNT MAX값 :",mns)
```

```
서울 AD1 COUNT 평균 개수 : 1.0203977600789558 서울 AD1 COUNT MAX값 : 23
서울 외 AD1 COUNT 평균 개수 : 1.0214825306893296 서울 AD1 COUNT MAX값 : 15
```

- 서울지역여부와 목표변수(COUNT)와의 상관관계
- 서울지역이 더 높은 COUNT값을 기록하는 경향이 존재

03 EDA & FEATURE ENGINEERING

```
# 강수여부 컬럼 생성
weekend['강수여부'] = 0
for i in range(len(weekend)):
    if weekend['강수량'].iloc[i] != 0.0:
        weekend['강수여부'].iloc[i] = 1
    if weekend['적설량'].iloc[i] != 0.0:
        weekend['강수여부'].iloc[i] = 1
```

```
# 이상 기온 여부 구분
everyday['이상기온'] =0
for i in range(len(everyday)):
    if everyday['일자'].iloc[i] in day:
        everyday['이상기온'].iloc[i] = 1
```

- **기상데이터를 활용해 일별 강수여부와 이상기온여부를 컬럼으로 생성(기상청 기준에 따름)**

03 EDA & FEATURE ENGINEERING

```
#강수여부와 목표변수와의 관계
df_rc0['COUNT'].value_counts()
```

```
1     778210
2      12284
3        973
4        237
5        113
6         58
7         30
9         12
8         11
10         6
15         4
11         3
12         3
13         3
23         2
22         1
17         1
19         1
20         1
Name: COUNT, dtype: int64
```

```
#강수여부와 목표변수와의 관계
df_rc1['COUNT'].value_counts()
```

```
1    85565
2     1235
3       87
4       34
5       20
7        6
6        5
Name: COUNT, dtype: int64
```

- 강수여부와 목표변수(COUNT)와의 상관관계
- 비가 오지 않은날에 더 높은 COUNT값을 기록하는 경향이 존재

03 EDA & FEATURE ENGINEERING

```
#이상기온여부와 목표변수와의 관계
df_stc0['COUNT'].value_counts()
```

```
1     847567
2      13251
3       1045
4        268
5        128
6         61
7         36
9         12
8         11
10         6
15         4
11         3
12         3
13         3
23         2
22         1
17         1
19         1
20         1
Name: COUNT, dtype: int64
```

```
#이상기온여부와 목표변수와의 관계
df_stc1['COUNT'].value_counts()
```

```
1    16208
2      268
3       15
5        5
4        3
6        2
Name: COUNT, dtype: int64
```

- 이상기온여부와 목표변수(COUNT)와의 상관관계
- 기온이 정상인 날에 더 높은 COUNT값을 기록하는 경향이 존재

03 EDA & FEATURE ENGINEERING 정리

(1). 평균적을 높은 COUNT 횟수를 기록하는 USER_ID 그룹이 존재
(2). 대체적으로 최근에 가입할수록 높은 COUNT 횟수를 기록
(3). D_TYPE은 타입별 분포의 특이성을 가짐(상관관계가 있다고 생각됨)
(4). STORE_ID는 AD1에 종속된 컬럼으로 AD10이 STORE_ID의 의미를 설명가능하다고 판단
(5). GOODS_TYPE은 타입별 분포의 특이성을 가짐(상관관계가 있다고 생각됨)
(6). DATE는 시계열 순으로 train과 test데이터를 나누었고 계절이나 연 순환 주기에 대한 판단이 어려움

03 EDA & FEATURE ENGINEERING 정리

(7). DATE를 주중,주말,공휴일로 나누자 주중보다는 휴일에 COUNT가 높은 횟수를 기록하는 상관관계를 보임
(8). AD1 지역이니셜로 추정, 지역을 서울과 비서울로 나누자 비서울그룹보다는 서울그룹에서 COUNT가 높은 횟수를 기록하는 상관관계를 보임
(9). 날씨데이터는 기상청의 정의에 따른 이상기후와 강수에 대해서만 일별로 유무를 체크해 컬럼으로 구성
(10). 전체 데이터가 평균적으로 서울 데이터의 날씨와 비슷한 날씨를 보일 것이라는 가설을 수립(데이터가 수도권에 많을 것이라고 가정)
(11). 날씨데이터는 타입별 분포의 특이성을 가짐(상관관계가 있다고 생각됨)
(12). 교통량데이터는 서울유입과 비서울로유출되는 교통량데이터로 나뉘는데 비서울로 유출되는 교통량데이터를 이용하기 힘들다.
(13). 일일단위 예측에서 분포의 특이성은 큰 영향이 없을 것으로 생각됨

04 예측모델링: 일별 COUNT 예측

AD1_TYPE_SEOUL	DAY_TYPE_주중	DAY_TYPE_주말	DAY_TYPE_공휴일	강수여부	이상기온	USER_ID_TYPE_A	USER_ID_TYPE_B	USER_ID_TYPE_C
1	0.00	0.00	1.00	0.00	0.00	1	0	0
1	0.00	0.00	1.00	0.00	0.00	1	0	0
1	0.00	0.00	1.00	0.00	0.00	1	0	0
0	0.00	0.00	1.00	0.00	0.00	1	0	0
1	0.00	0.00	1.00	0.00	0.00	1	0	0
...	...	...	...	...	...	...	...	...
1	1.00	0.00	0.00	0.00	1.00	1	0	0

- **Min-Max 알고리즘으로 정규화 이후 최종 Feature로 구성된 데이터셋 도출**

04 예측모델링: 일별 COUNT 예측

```
get_scores(models1, train_x1, train_y1, test_x1)
```

```
사용한 모델 : GradientBoostingRegressor()
MSE : 0.03745320059513653


사용한 모델 : XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.300000012, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=100, n_jobs=4, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
MSE : 0.038953190867117314


사용한 모델 : LGBMRegressor()
MSE : 0.036218743189760254


사용한 모델 : RandomForestRegressor()
MSE : 0.04461078021225606
```

- GB,XGB,LGBM, RF모델에 대하여 기본 모델로 정확도 평가
- 정확도가 가장 높은를 LGBM(light gbm)을 최종모델로 선정

04 예측모델링: 일별 COUNT 예측

```
# GridSearch로 가장 좋은 파라미터 찾기
GCV = GridSearchCV(model1, param_grid=param_grid, scoring='neg_mean_squared_error', cv=5, verbose=1, n_jobs=5)

# 모델 fitting
GCV.fit(train_x1, train_y1)

# 가장 좋은 파라미터 찾기
print("Best Param :", GCV.best_params_)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits

[Parallel(n_jobs=5)]: Using backend LokyBackend with 5 concurrent workers.
[Parallel(n_jobs=5)]: Done  40 tasks      | elapsed:  2.2min
[Parallel(n_jobs=5)]: Done  60 out of  60 | elapsed:  3.2min finished

Best Param : {'learning_rate': 0.05, 'max_depth': 10, 'n_estimators': 10, 'num_iterations': 100}
```

➢ **Grid Search로 최종파라미터 선정**

04 예측모델링: 일별 COUNT 예측

```
# 가장 좋은 파라미터로 모델 설정
model1=GCV.best_estimator_

# 예측
y_pred1 = model1.predict(test_x1)
```

```
print("MAE :",mean_absolute_error(real_count1, y_pred1))
print("MSE :",mean_squared_error(real_count1, y_pred1))
```

```
MAE : 0.04060358622564676
MSE : 0.03560501243087223
```

➢ **최종 모델을 통해 최종 정확도 산출**

04 예측모델링: 월별 COUNT 합계 예측

USER_A	USER_B	USER_C	1970	2013	2014	2015	2016	2017	2018	2019	2020	SEOUL	MONTH	COUNT	휴일	주말	주중	수여부	상기온	D_AA	D_BB	D_CC	G
1	0	0	0	0	0	0	0	0	0	0	1	1	10	1	1	0	0	0	0	1	0	0	
1	0	0	0	0	0	0	0	0	0	0	1	1	11	1	0	1	0	0	0	1	0	0	
1	0	0	0	0	0	0	0	0	0	0	1	1	12	1	0	1	0	0	0	1	0	0	
1	0	0	0	0	0	0	0	0	0	0	1	1	10	2	2	0	0	0	0	2	0	0	
1	0	0	0	0	0	0	0	0	0	0	1	1	10	2	1	0	1	0	0	0	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1	0	0	0	0	0	0	0	0	1	0	0	1	12	1	0	0	1	0	1	0	1	0	

- Min-Max 알고리즘으로 정규화 이후 최종 Feature로 구성된 데이터셋 도출
- 목표변수의 분포에 더 민감하게 Feature 구성

04 예측모델링: 월별 COUNT 합계 예측

```
rf_y_pred=rf.predict(df_test_x)
print("rf mse :",mean_squared_error(df_test_y,rf_y_pred))
print("rf mae :",mean_absolute_error(df_test_y,rf_y_pred))
gb_y_pred=gb.predict(df_test_x)
print("gb mse :",mean_squared_error(df_test_y,gb_y_pred))
print("gb mae :",mean_absolute_error(df_test_y,gb_y_pred))
lg_y_pred=lg.predict(df_test_x)
print("lg mse :",mean_squared_error(df_test_y,lg_y_pred))
print("lg mae :",mean_absolute_error(df_test_y,lg_y_pred))
```

```
rf mse : 0.3719841068114661
rf mae : 0.10656131371471773
gb mse : 0.2726050247630921
gb mae : 0.12493188245452637
lg mse : 0.26389459776750673
lg mae : 0.1039923412524817 5
```

- GB,LGBM,RF모델에 대하여 기본 모델로 정확도 평가
- 정확도가 가장 높은를 LGBM(light gbm)을 최종모델로 선정

04 예측모델링: 월별 COUNT 합계 예측

```
# GridSearch로 가장 좋은 파라미터 찾기
GCV = GridSearchCV(lg_model, param_grid=param_grid, scoring='neg_mean_squared_error', cv=5, verbose=1, n_jobs=5)
# 모델 fitting
GCV.fit(df_train_x, df_train_y)
# 가장 좋은 파라미터 찾기
print("Best Param :", GCV.best_params_)
# 가장 좋은 파라미터로 모델 설정
lg_model=GCV.best_estimator_
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
[Parallel(n_jobs=5)]: Using backend LokyBackend with 5 concurrent workers.
[Parallel(n_jobs=5)]: Done  40 tasks      | elapsed:  2.0min
[Parallel(n_jobs=5)]: Done  60 out of  60 | elapsed:  3.3min finished
Best Param : {'learning_rate': 0.05, 'max_depth': 20, 'n_estimators': 10, 'num_iterations': 200}
```

➢ **Grid Search로 최종파라미터 선정**

04 예측모델링: 월별 COUNT 합계 예측

```
#예측
lg_y_pred=lg_model.predict(df_test_x)
print('최종 mse(lightgbm모델 사용) : ',mean_squared_error(df_test_y,lg_y_pred))
print('최종 mae(lightgbm모델 사용) : ',mean_absolute_error(df_test_y,lg_y_pred))
```

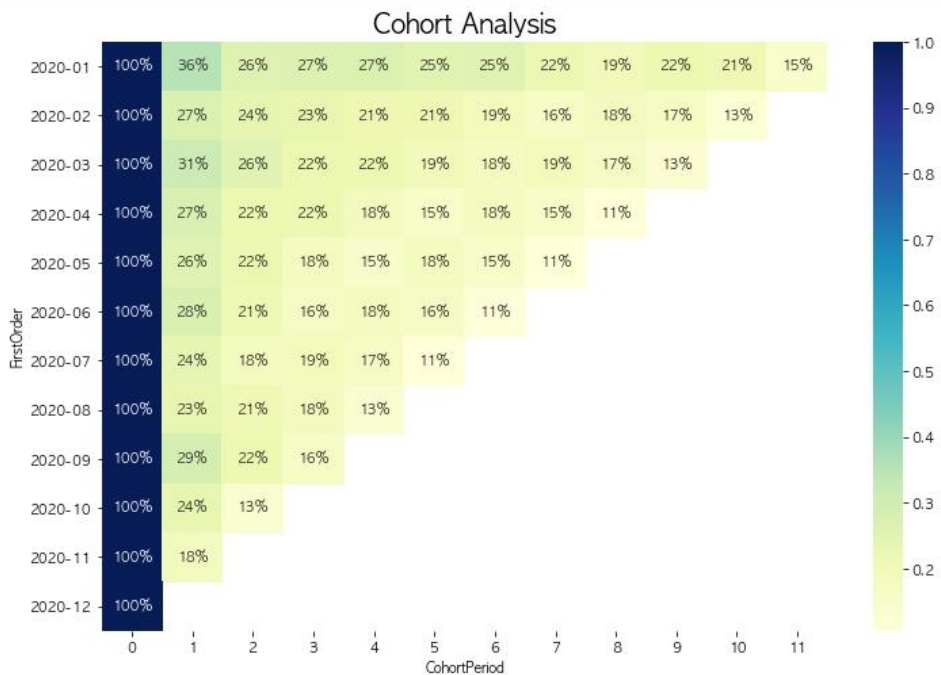
```
최종 mse(lightgbm모델 사용) :  0.26355187894719134
최종 mae(lightgbm모델 사용) :  0.10392207522470769
```

- **최종 모델을 통해 최종 정확도 산출**

05 코호트 분석

```
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
plt.figure(figsize=(12, 8))
plt.title('Cohort Analysis', fontsize=19)
sns.heatmap(user_retention, cmap ="YlGnBu", annot=True, fmt='.0%')
plt.show()
```



- 이용자 잔존율 (User retention)을 그래프로 시각화
- 분석결과 사용자 잔존율이 지속적으로 하락하고 있다는 사실 확인 가능
- Product Life Cycle이 좋은편이라고 할 수 없어 서비스의 개선이 필요

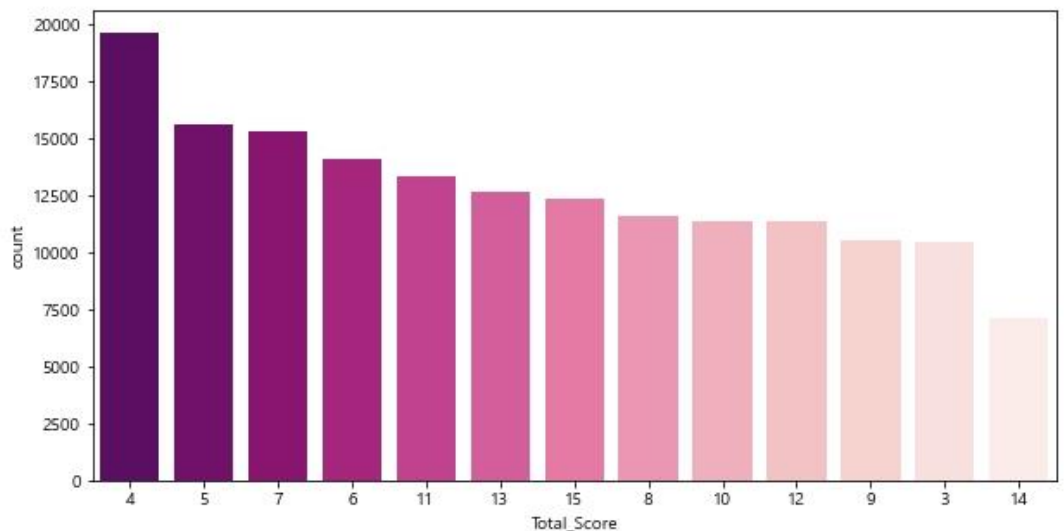
05 코호트 분석

- 코호트 분석결과 유저의 재사용률을 높이기 위해서 기존 서비스를 개선해야함
- 어플 평가를 분석해보았을 때 최신 정보 누락에 대한 불편함을 호소하는 유저가 많이 보이는 것으로 보아 어플 내 정보의 최신성을 유지하는 것이 중요
- 불법주차에 대한 불편함을 호소하는 유저가 많아 해당 부분에 대한 캠페인 진행 필요

06 RFM 분석

```
fig = plt.figure(figsize=(10,5))
sns.countplot(
    x='Total_Score',
    data=RFM_Result,
    order = RFM_Result['Total_Score'].value_counts().index,
    palette="RdPu_r")
```

```
<AxesSubplot:xlabel='Total_Score', ylabel='count'>
```



- Recency, Frequency, Monetary 세 가지 측면에서 고객을 평가
- 각각의 분류기준 별 1~5점으로 고객 평가
- 최종 합산 점수 분포를 시각화

06 RFM 분석

- RFM 분석결과 소수의 충성도 높은 고객 군집이 형성되어 있는 것으로 판단되니 해당 고객들을 대상으로 한 특별 고객관리가 요구됨
- 1회 사용유저로 추정되는 4점 유저들의 수가 굉장히 많은 것을 알 수 있으니 해당 고객들과 높은 점수를 받은 고객들을 비교 분석해 추가적인 마케팅 전략 수립이 요구됨