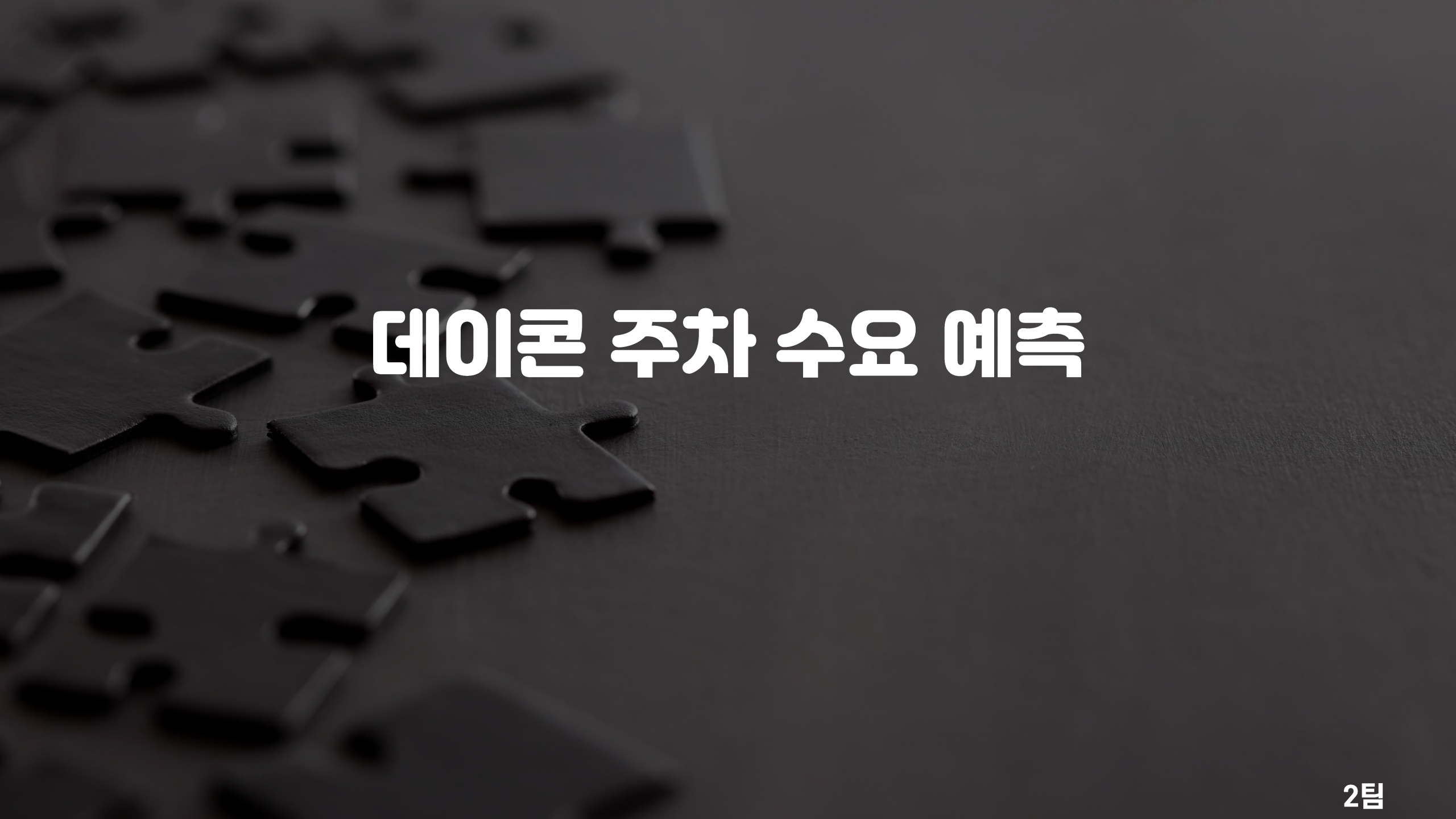


데이콘 주차 수요 예측

2팀

목차

데이터 설명

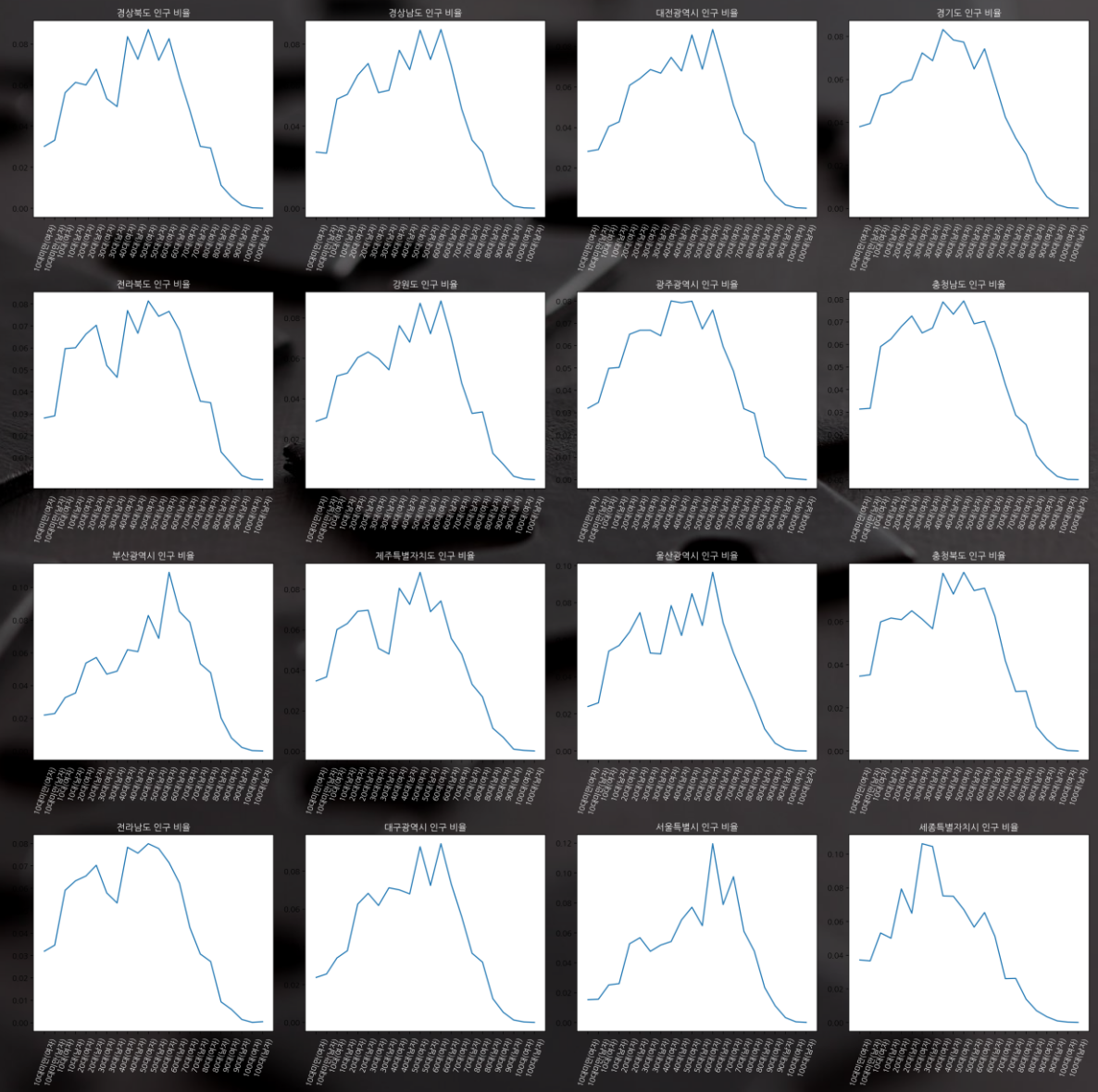
	단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	도보 10분거리 내 지하철역 수(환승노선 수 반영)	도보 10분거리 내 버스정류장 수	단지내주차면수	등록차량수
0	C2483	900	아파트	경상북도	국민임대	39.72	134	38.0	A	15667000	103680	0.0	3.0	1425.0	1015.0
1	C2483	900	아파트	경상북도	국민임대	39.72	15	38.0	A	15667000	103680	0.0	3.0	1425.0	1015.0
2	C2483	900	아파트	경상북도	국민임대	51.93	385	38.0	A	27304000	184330	0.0	3.0	1425.0	1015.0
3	C2483	900	아파트	경상북도	국민임대	51.93	15	38.0	A	27304000	184330	0.0	3.0	1425.0	1015.0
4	C2483	900	아파트	경상북도	국민임대	51.93	41	38.0	A	27304000	184330	0.0	3.0	1425.0	1015.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2947	C2532	239	아파트	강원도	국민임대	49.20	19	7.0	A	11346000	116090	0.0	1.0	166.0	146.0

train 데이터셋은 단지코드 별 아파트 혹은 상가 개개인의 특성 데이터가 있다. 예측해야 할 변수는 *등록차량수*로 단지코드내의 등록차량수는 같다.

	지역	10대미만(여자)	10대미만(남자)	10대(여자)	10대(남자)	20대(여자)	20대(남자)	30대(여자)	30대(남자)	40대(여자)	...	60대(여자)	60대(남자)	70대(여자)	70대(남자)	80대(여자)	80대
0	경상북도	0.030158	0.033195	0.056346	0.061360	0.060096	0.067859	0.053433	0.049572	0.083660	...	0.082684	0.063889	0.047717	0.030172	0.029361	0.011
1	경상남도	0.027400	0.026902	0.053257	0.055568	0.064920	0.070618	0.056414	0.057550	0.077092	...	0.087201	0.069562	0.048357	0.033277	0.027361	0.011
2	대전광역시	0.028197	0.029092	0.040490	0.042793	0.060834	0.064247	0.068654	0.066848	0.074667	...	0.088468	0.070261	0.051010	0.037143	0.032455	0.013
3	경기도	0.038030	0.039507	0.052546	0.053990	0.058484	0.059894	0.072331	0.068704	0.083208	...	0.074237	0.058419	0.042422	0.032725	0.025136	0.012

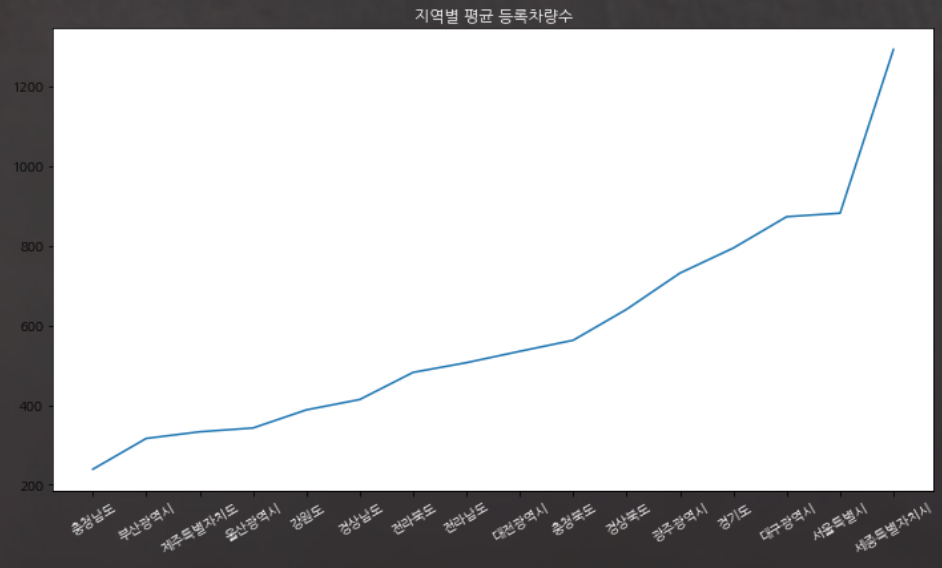
위 age_gender_info 데이터셋은 지역에 따른 인구분포를 보여준다. 지역별로 특정 연령대가 차지하는 비율로 나타냈다.

데이터 분석 – age_gender_info



age_gender_info 에서 지역에 따른 인구 비율 분포를 확인했을 때, 그래프의 모양이 비슷함을 알 수 있다. 서울이나, 부산이 비슷하고 그 외 지역은 20대 많고 40~50대가 많다.
다만, 세종시가 특별하게 20~30대가 많았다.

밑의 그래프로는 세종시가 평균 등록차량수가 많았다. 지역의 특성인지 나이대로 미루어봐도 20~30대,40~50대가 가진 차가 많아서 그런지 가설을 세워야 할 것 같다.



데이터 분석 – 자격유형 & 공급유형

train 데이터셋에서 *자격유형* 은 A가 가장 많고 O가 하나로 가장 적다.

처음엔 자격유형이 O인 아파트가 *공급유형*이 행복주택이라 행복주택만 봤는데 J K L M N O 밖에 없었다. 그리고 모두 행복주택이다. 행복주택의 자격유형이 J K L M N O를 맡는다.

*영구임대*는 대부분 C고, 영구임대와 국민임대를 제외한 공공임대(5년),10년,50년, 장기전세는 모두 A형이다.

test의 자격유형 2개가 비어있는데 C2411와 C2253다. 찾아보면 그냥 다 같은 유형이고 해서 각각 A와 C로 채워준다.

공공임대(5년, 10년, 50년, 분납)은 영구임대, 장기전세, 국민임대보다 평균적으로 *단지내주차면수*보다 *등록차량수*가 많다.

다만, 등록차량수가 많은지 단지내주차면수가 많은지 따졌을 때 자격유형의 차이는 없었다.

'B','F','O' 는 train 셋에만 있다. test에는 없고 train에만 있는 자격유형이다.

'공공분양', '공공임대(5년)', '장기전세' 는 test에 없고 train에만 있는 공급유형이다.

데이터 분석 – 결측값(임대료,임대보증금)

train이나 test나 임대료가 비어있으면 임대보증금도 비어있다.

train이나 test나 임대료가 비어있는 것의 자격유형은 D다.

'강원도', '경상남도', '대전광역시', '부산광역시', '제주특별자치도', '충청남도' 이 지역들이 임대료가 비어있는 것들이 있다.

*임대건물유형*을 봤는데 '상가'가 모두 임대료가 없다.

상가와 아파트가 껴있는 경우 등록차량수 보다 단지내주차면수가 많았다. 차를 많이 댈 수 있다는 뜻이다.

train과 test에는 '임대료'가 '-'인 경우도 있다.

그리고 임대보증금이 모두 비싸다. 임대료보다.

임대료 대비 보증금은 작게는 40배에서 크게는 1900배 정도로 분포해 있다. 평균 170배정도다.

데이터 분석 – 결측값(버스,지하철)

train 데이터 셋에서 *지하철*수가 1보다 크다면 대부분 단지내주차면수가 등록차량수보다 많다. 0에서는 비슷하다. 다시말해, 지하철수가 있다면 아파트에 보통 차 댈 수 있는수가 더 많다는 뜻이다.

*버스정류장*수는 6이나 8일경우 등록차량수가 더 많지만, 그 외는 아닌걸로 봐서 차이를 잘 모르겠다.

지하철은 대부분 없으니 0으로 채우는게 나아 보인다.

버스는 대부분 있어 보이니 중위값으로 채운다.

모델링 – 첫번째 시도.

```
df=train.loc[(train['임대보증금'].notnull())&(train['임대료']!='-')&(train['임대료'].notnull())]

df['임대료']=df['임대료'].astype(int)
df['임대보증금']=df['임대보증금'].astype(int)
```

임대료가 – 이거나 nan 값인 경우와 임대보증금이 nan 값인 경우 모두 제거

```
train['도보 10분거리 내 지하철역 수(환승노선 수 반영)']=train['도보 10분거리 내 지하철역 수(환승노선 수 반영)'].fillna(0)
train['도보 10분거리 내 버스정류장 수']=train['도보 10분거리 내 버스정류장 수'].fillna(0)

test['도보 10분거리 내 지하철역 수(환승노선 수 반영)']=test['도보 10분거리 내 지하철역 수(환승노선 수 반영)'].fillna(0)
```

나머지 빈값 0으로 채우기.

```
train=train.drop_duplicates()
test=test.drop_duplicates()

train=train.dropna()
test=test.dropna()
```

중복값 제거

```
train=train[['단지코드','총세대수','전용면적','단지내주차면수','등록차량수']]
test=test[['단지코드','총세대수','전용면적','단지내주차면수']]
```

숫자로만 변수 생성.

```
mean_area=pd.Series(train.groupby('단지코드')['전용면적'].mean(),name='mean면적')
median_area=pd.Series(train.groupby('단지코드')['전용면적'].median(),name='med면적')
std_area=pd.Series(train.groupby('단지코드')['전용면적'].std(),name='std면적')
min_area=pd.Series(train.groupby('단지코드')['전용면적'].min(),name='min면적')
max_area=pd.Series(train.groupby('단지코드')['전용면적'].max(),name='max면적')
sum_area=pd.Series(train.groupby('단지코드')['전용면적'].sum(),name='sum면적')

ts_mean_area=pd.Series(test.groupby('단지코드')['전용면적'].mean(),name='mean면적')
ts_median_area=pd.Series(test.groupby('단지코드')['전용면적'].median(),name='med면적')
ts_std_area=pd.Series(test.groupby('단지코드')['전용면적'].std(),name='std면적')
ts_min_area=pd.Series(test.groupby('단지코드')['전용면적'].min(),name='min면적')
ts_max_area=pd.Series(test.groupby('단지코드')['전용면적'].max(),name='max면적')
ts_sum_area=pd.Series(test.groupby('단지코드')['전용면적'].sum(),name='sum면적')
```

전용면적의 경우 세로로 여러 개 있는 데이터들을 가로로.
(aggregation 적용하여 mean,median,std,min,max,sum 값을 활용.)

후에 merge.

```
from sklearn.linear_model import LinearRegression
model=LinearRegression(normalize=False)
model.fit(train.drop('등록차량수',axis=1),train['등록차량수'])
pred=model.predict(test)
```

회귀 문제로 생각하여 예측.

결과 : 118.

모델링 최적화

예측값에서 -5 한 경우 리더보드결과가 제일 좋았다.

결과 : 115

모델링 – 두번째 시도.

```
df_train.loc[df_train.지역.isin(['서울특별시']), '지역'] = '이외'
df_train.loc[df_train.공급유형.isin(['공공임대(5년)', '장기전세', '공공분양']), '공급유형'] = '이외'
df_train.loc[df_train.자격유형.isin(['0', 'B', 'F']), '공급유형'] = '이외'
```

train에만 있었던 필드값들을 '이외'로 변환.

```
train.loc[(train.지역 == '충청남도') & (train['도보 10분거리 내 지하철역 수(환승노선 수 반영)'].isnull()), '도보 10분거리 내 지하철역 수
train.loc[(train.지역 == '경상남도') & (train['도보 10분거리 내 지하철역 수(환승노선 수 반영)'].isnull()), '도보 10분거리 내 지하철역 수
train.loc[(train.지역 == '대전광역시') & (train['도보 10분거리 내 지하철역 수(환승노선 수 반영)'].isnull()), '도보 10분거리 내 지하철역
```

충청남도,경상남도,대전광역시 지하철 없다.

```
train['임대건물구분'] = train['임대건물구분'].astype('category').cat.codes
train['지역'] = train['지역'].astype('category').cat.codes
train['공급유형'] = train['공급유형'].astype('category').cat.codes
train['자격유형'] = train['자격유형'].astype('category').cat.codes
```

범주형 변수들의 카테고리화.

```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
folds=[]
for train_idx, valid_idx in skf.split(train, train['등록차량수']):
    folds.append((train_idx, valid_idx))
```

stratifiedkfold로 5번으로 나눠 folds를 설정.

```
submit.iloc[:,1:]=0
pred_y = test['단지내주차면수']
pred_y = 0
for fold in range(5):
    pred_y += lgb_models[fold].predict(test_x)/5
```

```
result = pred_y.groupby('단지코드').sum()
value = pred_y.groupby('단지코드').count()

result = result/value
```

5번의 값을 5로 나눈 후 저장.
그 후, 개별 아파트 값을 예측 했으므로 단지코드로 그룹지어 데이터 개수를 직접 나눠준다.

결과 : 111

모델링 최적화

예측값에서 -30 하고 –값들을 모두 0으로 처리한 경우 리더보드결과가 제일 좋았다.

결과 : 108

추가 진행사항-1

최종 결과에서 동일하게 일부 값을 제거하게 LB 스코어가 상승되므로 예측 과정에서 사용 되는 값들의 크기를 줄이는 과정을 통해서 성능 향상을 이룰 수 있을 거라는 가정하에 진행

결과1. EDA 과정에서 확인되는 가장 큰 영향을 가지는 단지내 주차면수 값을 가공할 경우 훨씬 낮은 예측 결과를 얻게 됨. LB 3000

결과2. 그 외 다른 float 형 데이터들을 위와 같은 방식으로 변경하여 예측 진행 했을 때 처음 만든 Base 코드 대비 약간 더 낮은 성능 수치를 보임
LB 120~125

```
train['단지내주차면수비'] = train['단지내주차면수']*(train.전용면적별세대수/train.총세대수)
test['단지내주차면수비'] = test['단지내주차면수']*(test.전용면적별세대수/test.총세대수)
```