선형 모델

학습 목표

- 오류 처리된(2,3번 해결) 데이터를 활용한다.
- 데이터 EDA 및 시각화를 통해 데이터를 이해하고 기본 모델을 만들어본다.
- 모델을 제출해 본다.

- 데이콘 대회 : https://dacon.io/competitions/official/235745/overview/description (https://dacon.io/competitions/official/235745/overview/description)
- https://dacon.io/competitions/official/235745/talkboard/403708?page=1&dtype=recent (https://dacon.io/competitions/official/235745/talkboard/403708?page=1&dtype=recent)

01. 데이터 불러오기 및 확인

In [1]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
```

In [2]:

```
import pandas as pd

train = pd.read_csv("../data/parking_demand/train_df_errno.csv")
test = pd.read_csv("../data/parking_demand/test_df.csv")
sub = pd.read_csv("../data/parking_demand/sample_submission.csv")
age = pd.read_csv("../data/parking_demand/age_gender_info.csv")

train.shape, test.shape, sub.shape, age.shape
```

데이터 불러오기

Out[2]:

```
((2896, 15), (1008, 14), (150, 2), (16, 23))
```

In [3]:

```
train.columns
```

Out[3]:

```
Index(['단지코드', '총세대수', '임대건물구분', '지역', '공급유형', '전용면적', '전용
면적별세대수', '공가수',
       '자격유형', '임대보증금', '임대료', '10분내지하철수', '10분내버스정류장수',
'단지내주차면수', '등록차량수'],
      dtype='object')
```

In [4]:

```
train.columns = ['단지코드', '총세대수', '임대건물구분', '지역', '공급유형', '전용면적', '전용면적별
       '자격유형', '임대보증금', '임대료', '10분내지하철수',
       '10분내버스정류장수', '단지내주차면수', '등록차량수']

test.columns = ['단지코드', '총세대수', '임대건물구분', '지역', '공급유형', '전용면적', '전용면적별
       '자격유형', '임대보증금', '임대료', '10분내지하철수',
       '10분내버스정류장수', '단지내주차면수']
```

오류 데이터가 없는지 확인

In [5]:

```
train.loc[ ((train['단지코드']=='C2675') |
          (train['단지코드']=='C2335') |
          (train['단지코드']=='C1327') ) , :]
```

Out[5]:

단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	10분내지하철수	10분내버스정류장수	단지내주차면수	등록차량수

In [6]:

```
test.loc[ ((test['단지코드']=='C2675') |
          (test['단지코드']=='C2335') |
          (test['단지코드']=='C1327') ) , :]
```

Out[6]:

단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	10분내지하철수	10분내버스정류장수	단지내주차면수

오류 데이터 처리 확인

- ※ 동일한 단지에 코드가 2개로 부여된 단지 코드 (3쌍) : ['C2085', 'C1397'], ['C2431', 'C1649'], ['C1036', 'C2675']

- - (참고 사항) 주차면수는 하나의 단지임을 전제로 산정된 것이고 총세대수는 두 개 단지의 합계입니다. 다만 등록차량대수는 ['C2085', 'C1397'] 단지의 경우 동일 수치

In [7]:

```
# C2085, C1397 -> N2085
train.loc[ train['단지코드']=='N2085', : ].shape
```

Out[7]:

```
(14, 15)
```

오류 코드 변경

- C2431, C1649의 총세대수를 1047로 변경
- C2431, C1649의 등록차량대수를 1214로 변경
- C2431, C1649의 단지코드를 N2431로 변경

In [8]:

```
# C2431, C1649 -> N2431
print( train.loc[ train['단지코드']=='N2431', : ].shape )
train.loc[ train['단지코드']=='N2431', : ]
```

(6, 15)

Out[8]:

	단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	10분내지하철수	10분내버스정류장수	단지내주차면수	등
2293	N2431	1047	아파트	경상남도	공공임대(10년)	74.97	80	15.0	A	46000000	456000	NaN	NaN	1066.0	12
2294	N2431	1047	아파트	경상남도	공공임대(10년)	84.95	124	15.0	A	57000000	462000	NaN	NaN	1066.0	12
2295	N2431	1047	아파트	경상남도	공공임대(10년)	84.96	289	15.0	A	57000000	462000	NaN	NaN	1066.0	12
2296	N2431	1047	아파트	경상남도	공공임대(10년)	84.98	82	15.0	A	57000000	462000	NaN	NaN	1066.0	12
2350	N2431	1047	아파트	경상남도	국민임대	36.77	272	16.0	A	11217000	233330	0.0	2.0	1066.0	12
2351	N2431	1047	아파트	경상남도	국민임대	46.78	200	16.0	A	24389000	303220	0.0	2.0	1066.0	12

오류 코드 변경

- C1036의 총세대수를 1243로 변경
- C1036의 단지코드를 N1036로 변경

In [9]:

```
# C2085, C1397 -> N2085
train.loc[ train['단지코드']=='N1036', : ].shape
```

Out[9]:

(7, 15)

오류 3

3. 단지코드 등 기입 실수로 데이터 정제 과정에서 매칭 오류 발생
 - (오류 내용) 단지코드 등 기입 실수로 총세대수가 주차면수에 비해 과하게 많거나 적은 경우가 발생하였고, 점검 결과 일부 데이터의 단지코드, 총세대수, 주차면수 등에서 오류가 검출되었습니다.
 - (발생 원인) 원천데이터 수집 과정에서 단지 코드 등이 잘못 기입되었고 이를 인지하지 못한 채 데이터 정제를 하여 오류가 발생하였습니다.
 - (관련 데이터) 아래와 같이 총 9개 단지에서 같은 문제가 확인되었습니다.
※ 실수가 발생한 단지 코드 (9개 단지) : ['C2335', 'C1327', 'C1095', 'C2051', 'C1218', 'C1894', 'C2483', 'C1502', 'C1988']
 - C2335, C1327 단지는 테스트셋, 나머지는 트레인셋 입니다.

오류 처리

- train 데이터 셋에 오류 발생 코드를 ERR04로 변경 후, 데이터 셋을 두개로 분리

In [10]:

```
train.loc[ train['단지코드'].str.contains('ERR'), :].shape
```

Out[10]:

(0, 15)

02. 결측치를 처리(1)

In [11]:

```
train.isnull().sum()
```

Out[11]:

```
단지코드              0
총세대수              0
임대건물구분            0
지역                0
공급유형              0
전용면적              0
전용면적별세대수          0
공가수               0
자격유형              0
임대보증금           569
임대료             569
10분내지하철수        211
10분내버스정류장수       4
단지내주차면수           0
등록차량수             0
dtype: int64
```

In [12]:

```
test.isnull().sum()
```

Out[12]:

```
단지코드              0
총세대수              0
임대건물구분            0
지역                0
공급유형              0
전용면적              0
전용면적별세대수          0
공가수               0
자격유형              2
임대보증금           180
임대료             180
10분내지하철수         38
10분내버스정류장수       0
단지내주차면수           0
dtype: int64
```

결측치 처리를 어떻게 할 것인가?

자격유형(test) 결측치 처리

In [13]:

```
train['지역'].unique()
```

Out[13]:

```
array(['경상남도', '대전광역시', '경기도', '전라북도', '강원도', '광주광역시', '충청
남도', '부산광역시',
       '제주특별자치도', '울산광역시', '충청북도', '전라남도', '경상북도', '대구광역
시', '서울특별시',
       '세종특별자치시'], dtype=object)
```

In [14]:

```
test['지역'].unique()
```

Out[14]:

```
array(['경기도', '부산광역시', '전라북도', '경상남도', '충청남도', '대전광역시', '제
주특별자치도', '강원도',
       '울산광역시', '경상북도', '충청북도', '광주광역시', '전라남도', '대구광역시',
'세종특별자치시'],
      dtype=object)
```

In [15]: 어떤것이 비어있을까?

```
test.loc[test['자격유형'].isnull()]
```

Out[15]:

이구분을활용해서 자격 유형의 결측값을 찾을 수 있지 않을까?

	단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	10분내지하철수	10분내버스정류장수	단지내주차면수
196	C2411	962	아파트	경상남도	국민임대	46.90	240	25.0	NaN	71950000	37470	0.0	2.0	840.0
258	C2253	1161	아파트	강원도	영구임대	26.37	745	0.0	NaN	2249000	44770	0.0	2.0	173.0

In [16]: 해당되는 구분으로 정보를 가져와 보기.

```
grouped = test.groupby(['단지코드', '임대건물구분', '지역','공급유형'])
group1 = grouped.get_group( ('C2411', '아파트', '경상남도', '국민임대')  )
group2 = grouped.get_group( ('C2253', '아파트', '강원도', '영구임대')  )
group2
```

Out[16]:

아마도 자격유형은 C가 아닐까?

	단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	10분내지하철수	10분내버스정류장수	단지내주차면수
258	C2253	1161	아파트	강원도	영구임대	26.37	745	0.0	NaN	2249000	44770	0.0	2.0	173.0
259	C2253	1161	아파트	강원도	영구임대	31.32	239	0.0	C	3731000	83020	0.0	2.0	173.0
260	C2253	1161	아파트	강원도	영구임대	31.32	149	0.0	C	3731000	83020	0.0	2.0	173.0

In [17]:

```
test.loc[ 196, "자격유형"] = 'A'
test.loc[ 258, "자격유형"] = 'C'
```

In [18]:

```
train.head(3)
```

Out[18]:

	단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	10분내지하철수	10분내버스정류장수	단지내주차면수	등록차량수
0	C2515	545	아파트	경상남도	국민임대	33.48	276	17.0	A	9216000	82940	0.0	3.0	624.0	205.0
1	C2515	545	아파트	경상남도	국민임대	39.60	60	17.0	A	12672000	107130	0.0	3.0	624.0	205.0
2	C2515	545	아파트	경상남도	국민임대	39.60	20	17.0	A	12672000	107130	0.0	3.0	624.0	205.0

In [19]:

```
test.head(3)
```

Out[19]:

	단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	10분내지하철수	10분내버스정류장수	단지내주차면수
0	C1072	754	아파트	경기도	국민임대	39.79	116	14.0	H	22830000	189840	0.0	2.0	683.0
1	C1072	754	아파트	경기도	국민임대	46.81	30	14.0	A	36048000	249930	0.0	2.0	683.0
2	C1072	754	아파트	경기도	국민임대	46.90	112	14.0	H	36048000	249930	0.0	2.0	683.0

In [20]: 자격유형은 train, test는 어떤 것이 있을까?

```
print(train.자격유형.unique())
print(test.자격유형.unique())
```

```
['A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L' 'M' 'N' 'O']
['H' 'A' 'E' 'C' 'D' 'G' 'I' 'J' 'K' 'L' 'M' 'N']
```

train에는 있고, test는 없는것이 있고, test에는 있고, train에 없는 것이 있다.

우리가 예측하는 것이 test의 자격유형만 있다면

...

In [21]: 우선은 train, test에 있는 친구들 전부를 mapping을 함.

```
mapping = { 'A':1, 'B':2, 'C':3, 'D':4, 'E':5,
            'F':6, 'G':7, 'H':8, 'I':9, 'J':10,
            'K':11, 'L':12, 'M':13, 'N':14, 'O':15  }

train['자격유형'] =train['자격유형'].map(mapping).astype(int) 만약에 전체를 변경했다.
test['자격유형'] =test['자격유형'].map(mapping).astype(int)

train.head(3)
```

Out[21]:

	단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	10분내지하철수	10분내버스정류장수	단지내주차면수	등록차량수
0	C2515	545	아파트	경상남도	국민임대	33.48	276	17.0	1	9216000	82940	0.0	3.0	624.0	205.0
1	C2515	545	아파트	경상남도	국민임대	39.60	60	17.0	1	12672000	107130	0.0	3.0	624.0	205.0
2	C2515	545	아파트	경상남도	국민임대	39.60	20	17.0	1	12672000	107130	0.0	3.0	624.0	205.0

In [22]:

```
test.head(3)
```

Out[22]:

	단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	10분내지하철수	10분내버스정류장수	단지내주차면수
0	C1072	754	아파트	경기도	국민임대	39.79	116	14.0	8	22830000	189840	0.0	2.0	683.0
1	C1072	754	아파트	경기도	국민임대	46.81	30	14.0	1	36048000	249930	0.0	2.0	683.0
2	C1072	754	아파트	경기도	국민임대	46.90	112	14.0	8	36048000	249930	0.0	2.0	683.0

In [23]:

```
print(train.공급유형.unique())
print(test.공급유형.unique())
```

```
['국민임대' '공공임대(50년)' '영구임대' '임대상가' '공공임대(10년)' '공공임대(분납)'
'장기전세' '공공분양'
 '행복주택' '공공임대(5년)']
['국민임대' '영구임대' '임대상가' '공공임대(50년)' '공공임대(10년)' '공공임대(분납)'
'행복주택']
```

In [24]:

```
train.columns
```

Out[24]:

```
Index(['단지코드', '총세대수', '임대건물구분', '지역', '공급유형', '전용면적', '전용
면적별세대수', '공가수',
       '자격유형', '임대보증금', '임대료', '10분내지하철수', '10분내버스정류장수',
'단지내주차면수', '등록차량수'],
      dtype='object')
```

10분내버스정류장수 (tr) 결측치 처리,

In [25]:

```
train.isnull().sum()
```

Out[25]:

```
단지코드              0
총세대수              0
임대건물구분            0
지역                0
공급유형              0
전용면적              0
전용면적별세대수          0
공가수               0
자격유형              0
임대보증금           569
임대료             569
10분내지하철수        211
10분내버스정류장수       4
단지내주차면수           0
등록차량수             0
dtype: int64
```

In [26]:

```
train.loc[ train['10분내버스정류장수'].isnull(), :]
```

Out[26]:

	단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	10분내지하철수	10분내버스정류장수	단지내주차면수	등
2293	N2431	1047	아파트	경상남도	공공임대(10년)	74.97	80	15.0	1	46000000	456000	NaN	NaN	1066.0	12
2294	N2431	1047	아파트	경상남도	공공임대(10년)	84.95	124	15.0	1	57000000	462000	NaN	NaN	1066.0	12
2295	N2431	1047	아파트	경상남도	공공임대(10년)	84.96	289	15.0	1	57000000	462000	NaN	NaN	1066.0	12
2296	N2431	1047	아파트	경상남도	공공임대(10년)	84.98	82	15.0	1	57000000	462000	NaN	NaN	1066.0	12

In [27]:

```
grouped = train.groupby(['임대건물구분', '지역','공급유형', '자격유형'])
group1 = grouped.get_group( ('아파트', '경상남도', '공공임대(10년)', 1)  )
group1
```

Out[27]:

	단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	10분내지하철수	10분내버스정류장수	단지내주차면수	등록차량수
2158	C1788	376	아파트	경상남도	공공임대(10년)	51.59	116	28.0	1	29000000	340000	0.0	3.0	380.0	412.0
..	..	..	..	경	공										

In [28]: 수십개의 데이터를 보면서... 아마 4가 아닐까?

```
# 데이터 확인 후, 임의 처리 4
train.loc[ train['10분내버스정류장수'].isnull(), "10분내버스정류장수"] = 4
```

In [29]:

```
train.loc[ train['10분내버스정류장수'].isnull(), :]
```

Out[29]:

단지코드	총세대수	임대건물구분	지역	공급유형	전용면적	전용면적별세대수	공가수	자격유형	임대보증금	임대료	10분내지하철수	10분내버스정류장수	단지내주차면수	등록차량수

In [30]:

```
train.corr()['등록차량수']
```

Out[30]:

```
총세대수            0.333440
전용면적            0.112717
전용면적별세대수        0.250513
공가수             0.118910
자격유형            -0.154482
10분내지하철수       -0.107308
10분내버스정류장수      0.104203
단지내주차면수         0.861338
등록차량수           1.000000
Name: 등록차량수, dtype: float64
```

In [31]:

```
from sklearn.linear_model import LinearRegression
import numpy as np
from sklearn.model_selection import train_test_split
```

In [34]: 컬럼을 feature를 여러개 선택해 보기.

```
sel = ['총세대수', '전용면적', '전용면적별세대수',
       '공가수',  '단지내주차면수', '자격유형']

X = train[sel]
y = train['등록차량수']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=0)
```

In [35]: 회귀 모형 만들기.

```
model = LinearRegression()
model.fit(X_train, y_train)
pred = model.predict(X_test)
```

In [36]: 결정계수 점수 확인.

```
print("학습(score) :", model.score(X_train, y_train) ) # 결정계수
print("테스트(score) :", model.score(X_test, y_test) ) # 결정계수
```

```
학습(score) : 0.7848278438379498
테스트(score) : 0.7841937560157191
```

학습과 테스트가 거의 차이가 없다.

아 분포가 거의 비슷한 걸까?
아니면 우리가 잘 예측한 것일까?

In [37]:

```
# mae
np.mean( np.abs(y_test - pred)  )
```

Out[37]:

```
147.8977567932893
```

In [38]:

```
# mse
mse_val = np.mean( (y_test - pred)**2  )
mse_val
```

Out[38]:

```
43640.968133126094
```

In [39]:

```
# rmse
np.sqrt(  mse_val )
```

Out[39]:

```
208.9042080311598
```