SNU 빅데이터 핀테크 과정 기계학습딥러닝

김해시 화재발생 예측모델 개발

권예준, 김경빈, 원소현, 이아연, 전우근

INDEX

01

과제 개요

COMPAS

COMPAS

나침반(Compass)과 같이 도시문제 해결의
방향성을 제시하는 데이터 분석 플랫폼

- **C** Citizen, Crowdsourcing
- **O** Opportunity, Offer
- **M** Management
- **P** Problem, Public
- **A** Analysis, AI, Advance
- **S** Solution, Share, Sustainable

배경

김해화재 발생 기름덩어리 양산까지 날아와

[종합] 경남 김해 공장 화재로 건물 2동 전소...인명피해는 없는 듯..

경남 김해 기계부품창고 화재...4시간만에 진화

[김해] 5일 새벽 폐기물 처리공장 화재

김해 구산동 아파트 6층서 화재...어린이 1명 사망·엄마 화상

김해지역은 화재가 계절 및 장소 등에 관계 없이 잇따라 발생하고 있음

목적

김해시에서는 소방 및 건물관련 정보를 융합하여 **지역 내 화재 위험도**에 대해 분석 및 예측하고, 이를 이용하여 화재에 대한 집중적이고 적극적인 예방활동을 수행함으로써 행정력의 효율적인 배분에 기여하고자 함.

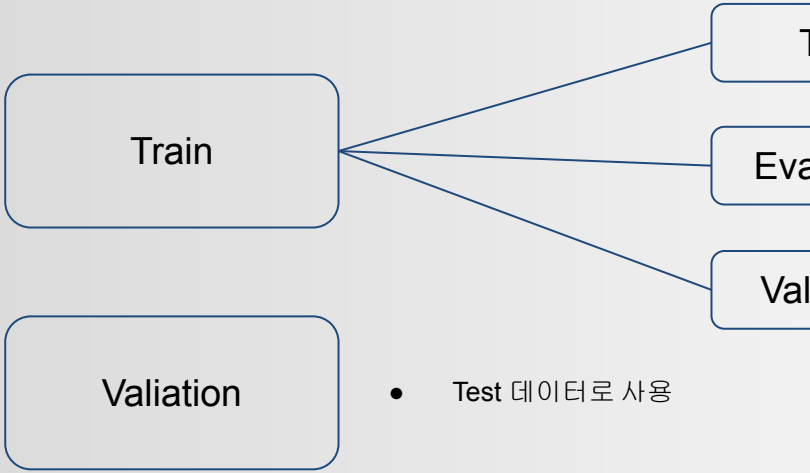
해결 과제

김해시가 수집한 소방 및 건물 관련 데이터를 활용하여 건축물의 화재 위험도 분석 및 예측 모델 제시

02

데이터 전처리

0) 데이터



- 0.6, 학습 데이터
- 0.2, 조기 종료를 위한 비교데이터
- 0.2, 학습된 모델 평가용
- Test 데이터로 사용
- 화재 발생 여부 정보 x
- 학습 & 검증에 쓸 수 없음

1) 데이터 구조파악

> 총 179개의 column 과 59199개의 데이터로 구성되어 있다.

	dt_of_fr	fr_yn	bldng_us	bldng_archtctr	bldng_cnt	bldng_ar	ttl_ar	lnd_ar	dt_of_athrztn	ttl_grnd_flr	ttl_dwn_flr	bldng_us_clssfctn	tmprtr	prcpttn	wnd_spd	wnd_drctn	hmdt	gas_engry_us_201401
0	2017-10-20 05:54	Y	단독주택	블록구조	3	69.420	69.4200	0.0	1977	1.0	0.0	주거용	10.7	NaN	0.9	200.0	96.0	0.0
1	2018-09-30 08:26	N	NaN	NaN	3	46.290	46.2900	0.0	NaN	1.0	0.0	NaN	19.5	NaN	0.5	20.0	74.0	NaN
2	2016-10-30 14:57	Y	공동주택	철근콘크리트구조	1	583.800	2516.7600	1446.0	2.00011e+07	5.0	0.0	주거용	15.5	NaN	2.0	90.0	21.0	0.0
3	2016-06-14 05:23	N	단독주택	일반목구조	2	48.920	48.9200	0.0	1936	1.0	0.0	주거용	20.6	NaN	0.4	0.0	91.0	0.0
4	2018-04-22 05:38	N	NaN	NaN	2	0.000	0.0000	0.0	NaN	NaN	NaN	NaN	12.7	NaN	0.6	360.0	89.0	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
59194	2016-12-03 14:25	N	동·식물 관련시설	경량철골구조	2	423.000	423.0000	3010.0	1.99612e+07	NaN	NaN	NaN	11.2	NaN	0.2	0.0	41.0	0.0
59195	2016-10-29 10:10	N	NaN	NaN	35	5593.711	54835.0382	0.0	NaN	171.0	0.0	NaN	14.8	NaN	1.9	20.0	72.0	NaN
59196	2018-03-03 19:49	N	NaN	NaN	2	2436.820	2468.8000	6512.0	NaN	4.0	0.0	NaN	13.3	NaN	3.2	230.0	58.0	NaN
59197	2017-09-27 11:10	N	제1종근린생활시설	경량철골구조	2	283.200	207.7100	0.0	2.00512e+07	2.0	0.0	상업용	18.2	1.0	1.4	360.0	95.0	0.0
59198	2015-09-20 07:34	N	창고시설	경량철골구조	3	169.260	169.2600	537.0	2.00205e+07	1.0	0.0	기타	16.5	NaN	0.8	200.0	91.0	0.0

59199 rows × 179 columns

1) 데이터 구조파악

Column(179개) :

화재발생일시, 화재발생여부, 건물용도, 건물구조, 건물채수, 건물건축면적, 건물연면적, 토지면적, 건물승인일자, 건물들의 지상층수의 합, 건물들의 지하 층수의 합, 건물용도분류명, 온도(c), 강수량, 풍속, 풍향, 습도, 전기 에너지 사용량(2014년1월~2018년 12월), 가스사용량 (2014.01 ~ 2014.01), 복도/계단/출입구의 성능 유지여부(0~5), 옥상광장의 피난성능 유지여부(0~5), 방화문/방화셔터 등의 성능, 유지여부(0~5) , 방화구획 적합 여부(0~5) , 경계벽 및 칸막이벽의 변경 등 방화성능 유지여부 (0~5), 배연설비의 성능 유지여부(0~5), 내화구조의 성능 유지여부(0~5), 방화벽의 성능 유지여부 (0~5), 외벽의 성능 유지여부(0~5), 창호의 성능 유지여부(0~5)

계절적 특징이 나타날 것으로 예상되는 전기사용량과 가스사용량은 월평균으로 합침

1) 데이터 구조파악

Column 명(179개) :

내부마감의 방화성능 유지여부(0~5), 외부마감의 노후화 및 마감재 탈락 여부(0~5), 지하층의 소방설비 성능 유지여부(0~5), 지하층 피난구/피난계단의 성능 유지여부(0~5), 지적상 지목, 용도지역지구명, 용도지역지구명2, 토지이용상황명, 도로측면명, 행정구역명, 행정구역 인구, 119 안전센터와의 거리, 단위 면적당 건물 가격(2019년), 소방용수시설(소화전 등)과의 최소 거리, 관할 소방서 인원, 다중이용시설 포함여부, 공공 CCTV와의 최소 거리, 반경 100m 이내 공공 CCTV, 반경 100m 이내 소방용수 시설 수, 담배 소매점과의 최소 거리, 안전 비상벨과의 최소 거리, 자동 심장 충격기와의 최소 거리, 금연구역과의 최소 거리, 소방관리대상물기준, 소방시설특례5호여부, 소방시설특례6호여부, 사용여부, 위험물대상여부, 자체소방대여부, 대량위험물제조소등여부, 문화재여부

2) 변수 처리

> 2014년~2018년동안 사용된 전기/가스량을 월별 평균으로 변경

```
# 201401~201801 까지 5개년을 더해서 ele_engry_us_01 을 만듬, 이때 기록 있으면 count 증가해서 평균 만듦

gas_energy_us_m = []
ele_energy_us_m = []
for i in range(len(df_train2)):
  j = 0
  months_g = []
  months_e = []
  for m in range(12):
    count_g = 0
    rec_value_g = 0
    count_e = 0
    rec_value_e = 0
    for y in range(5):
      if df_train2.iloc[i][gas_energy_us_month[m][y]] > 0 :
        rec_value_g += df_train2.iloc[i][gas_energy_us_month[m][y]]
        count_g += 1
      if df_train2.iloc[i][ele_energy_us_month[m][y]] > 0 :
        rec_value_e += df_train2.iloc[i][ele_energy_us_month[m][y]]
        count_e += 1

    if count_g == 0:
      months_g.append(0)
    else:
      months_g.append(rec_value_g/count_g)

    if count_e == 0:
      months_e.append(0)
    else:
      months_e.append(rec_value_e/count_e)

  gas_energy_us_m.append(months_g)
  ele_energy_us_m.append(months_e)
```

```
print("전처리 전 크기",df_train.shape)
print("전처리 후 크기",df_train_tmp.shape)
```

```
전처리 전 크기 (59199, 179)
전처리 후 크기 (59199, 84)
```

120 > 24로 96개 column감소

2) 변수 처리

> 승인날짜 건물 승인 여부로 가정, 결측치 처리 : (건물) 있음: 1, 없음: 0

```
df_train.dt_of_athrztn.fillna(123456,inplace=True)
df_train.dt_of_athrztn = np.where(df_train.dt_of_athrztn == 123456,0,1)
df_train.loc[(df_train['dt_of_athrztn']==0) & (df_train['bldng_us'].isnull()==True), 'bldng_us'] = '미확인'
df_train.bldng_us.fillna(0,inplace=True)
```

승인날짜

dt_of_athrztn
1.0
0.0
1.0
1.0
0.0
...
1.0
0.0
0.0
1.0
1.0

> train set 과 test set 동시에 적용 불가능한 변수 제거

```
df_train.drop('행정구역명', axis=1, inplace=True)
df_train.drop('도로측면명', axis=1, inplace=True)
```

> 기타 변수 제거

```
df_train.drop(['dt_of_fr',
 '소방시설특례5호여부',
 '소방시설특례6호여부',
 '자체소방대여부'], axis=1, inplace=True)
```

3) 결측치 처리 ver.1

> 건물 용도별 결측치 처리 :

Ordinal & 카테고리 → 최빈치 / Numerical → 평균

```
    # ordinal_cat
    for column in ordinal_cat:
        val_cnt = temp_data[column].value_counts()
        if len(val_cnt) != 0:
            freq = val_cnt.idxmax()
#most_freq.append(freq)
        elif len(val_cnt) == 0:
            freq = 0
        temp_data[column].fillna(freq,inplace=True)
```

```
    # numerical_cat
    for column in numerical_cat:
        val_cnt = temp_data[column].value_counts()
        if len(val_cnt) != 0:
            mean = temp_data[column].mean()
        elif len(val_cnt) == 0:
            mean = 0

        temp_data[column].fillna(mean,inplace=True)
```

	건물용도 bldng_us	승인날짜 dt_of_athrztn
0	단독주택	1.0
1	미확인	0.0
2	공동주택	1.0
3	단독주택	1.0
4	미확인	0.0
...	...	...
59194	동.식물 관련시설	1.0
59195	미확인	0.0
59196	미확인	0.0
59197	제1종근린생활시설	1.0
59198	창고시설	1.0

3) 결측치 처리 ver.1

> Categorical : **OneHotEncoding**

```
cat_col_idx = np.where(X.dtypes == object)[0]
cat_col_list =[]

for idx in cat_col_idx:
    col = X.columns[idx]
    cat_col_list.append(col)
    temp_data = pd.get_dummies(X[col],prefix="{}_".format(col))
    X = X.merge(temp_data,left_index=True,right_index=True)

X.drop(columns=cat_col_list,inplace=True)
```

```
X.shape
```

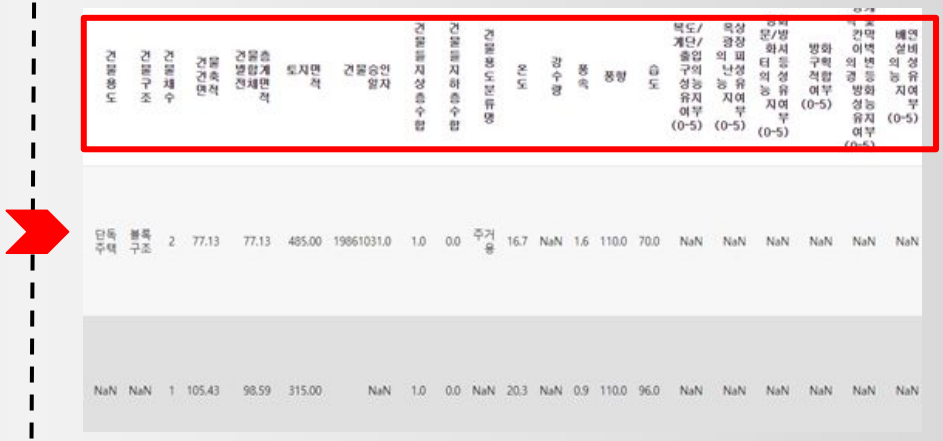
```
(59199, 252)
```

Scaling : **Robust Scaler**

4) 결측치 처리 ver.2

> Column renaming
한글로 열의 명칭 전환

bldng_cnt	bldng_ar	ttl_ar	lnd_ar	dt_of_athrztn	ttl_grnd_flr	ttl_dwn_flr	bldng_us_clssfctn	tmprtr	prcpttn	wnd_spd	wnd_drctn	hmdt
2	77.13	77.13	485.0	19861031.0	1.0	0.0	주거용	16.7	NaN	1.6	110.0	70.0
1	105.43	98.59	315.0	NaN	1.0	0.0	NaN	20.3	NaN	0.9	110.0	96.0



4) 결측치 처리 ver.2

> Imputation

'iterativeimputer' 함수를 사용하여 다른 변수들과의 상관성을 고려해 nan값을 채움

```
imputer = IterativeImputer(ExtraTreesRegressor(n_estimators=3,max_depth=5), n_nearest_features=3, max_iter=5, random_state=11)
encode_data = pd.DataFrame(np.round(imputer.fit_transform(df_train)),columns = df_train.columns)
```

	건물용도	건물구조	건물체수	건물건면적	건물층별계체전면적	토지면적	건물승인여부	건물들지상층수합	건물들지하층수합	건물용도분류명	온도	강수량	풍속	풍향	습도	복도/계단/출입구의 성능유지 여부 (0-5)	옥상 광장의 난성능 지여부 (0-5)	방화문/방화셔터의 능 성유지여부 (0-5)	방화구획적합여부 (0-5)	및 칸막이벽의 경 방화성능유지 여부 (0-5)	배연설비의 성능유지여부 (0-5)	내화구조의 성능유지여부 (0-5)	방화벽의 성능유지여부 (0-5)	외벽의 성능유지여부 (0-5)	창호의 성능유지여부 (0-5)	내부마감의 방화성능 유지여부 (0-5)
0	7.0	8.0	2.0	77.0	77.0	485.0	1.0	1.0	0.0	6.0	17.0	2.0	2.0	110.0	70.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	2.0	0.0	1.0	3.0
1	8.0	9.0	1.0	105.0	99.0	315.0	0.0	1.0	0.0	5.0	20.0	2.0	1.0	110.0	96.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	2.0	0.0	1.0	3.0
2	7.0	15.0	1.0	118.0	290.0	197.0	1.0	3.0	0.0	6.0	2.0	2.0	1.0	320.0	68.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	2.0	0.0	1.0	3.0
3	1.0	11.0	1.0	336.0	336.0	1360.0	1.0	1.0	0.0	1.0	6.0	2.0	3.0	290.0	32.0	3.0	3.0	3.0	3.0	3.0	1.0	3.0	2.0	0.0	1.0	3.0
4	7.0	7.0	3.0	251.0	250.0	840.0	1.0	1.0	0.0	6.0	16.0	2.0	1.0	360.0	80.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	2.0	0.0	1.0	3.0

4) 결측치 처리 ver.2

> One-hot-encoding

모든 범주형 데이터 값들을 숫자형으로 변환

```
# create a list of categorical columns to iterate over
encoder = OrdinalEncoder()

def encode(data):
    '''function to encode non-null data and replace it in the original data'''
    #retains only non-null values
    nonulls = np.array(data.dropna())
    #reshapes the data for encoding
    impute_reshape = nonulls.reshape(-1,1)
    #encode date
    impute_ordinal = encoder.fit_transform(impute_reshape.astype(str))
    #Assign back encoded values to non-null values
    data.loc[data.notnull()] = np.squeeze(impute_ordinal)
    return data

#create a for loop to iterate through each column in the data
for columns in cat:
    encode(df_train[columns])
```

```
new_train = pd.get_dummies(encode_data, columns =cat)
```

```
new_train
```

	건물 층수	건물 건축 면적	건물 총괄 합계 전체 면적	토지 면적	건물 승인 여부	건물 들지 상층 수합	건물 들지 하층 수합	온도	강수 량	풍속	풍향	습도	복도/계단/출입구의 성능 유지 여부 (0-5)	옥상 광장 의 난성 능 유지 여부 (0-5)	방화문/방화셔터의 능 성 유지 여부 (0-5)	방화 구획 적합 여부 (0-5)	경계벽 및 칸막이벽 의 변경 등 방화 성능 유지 여부 (0-5)	배연 설비 의 성 능 유지 여부 (0-5)	내 의 능 (0
0	2.0	77.0	77.0	485.0	1.0	1.0	0.0	17.0	2.0	2.0	110.0	70.0	3.0	3.0	3.0	3.0	3.0	2.0	
1	1.0	105.0	99.0	315.0	0.0	1.0	0.0	20.0	2.0	1.0	110.0	96.0	3.0	3.0	3.0	3.0	3.0	2.0	
2	1.0	118.0	290.0	197.0	1.0	3.0	0.0	2.0	2.0	1.0	320.0	68.0	3.0	3.0	3.0	3.0	3.0	2.0	
3	1.0	336.0	336.0	1360.0	1.0	1.0	0.0	6.0	2.0	3.0	290.0	32.0	3.0	3.0	3.0	3.0	3.0	1.0	
4	3.0	251.0	250.0	840.0	1.0	1.0	0.0	16.0	2.0	1.0	360.0	80.0	3.0	3.0	3.0	3.0	3.0	2.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	

4) 결측치 처리 ver.2

> Categorical 변수 중 결측치 개수 파악

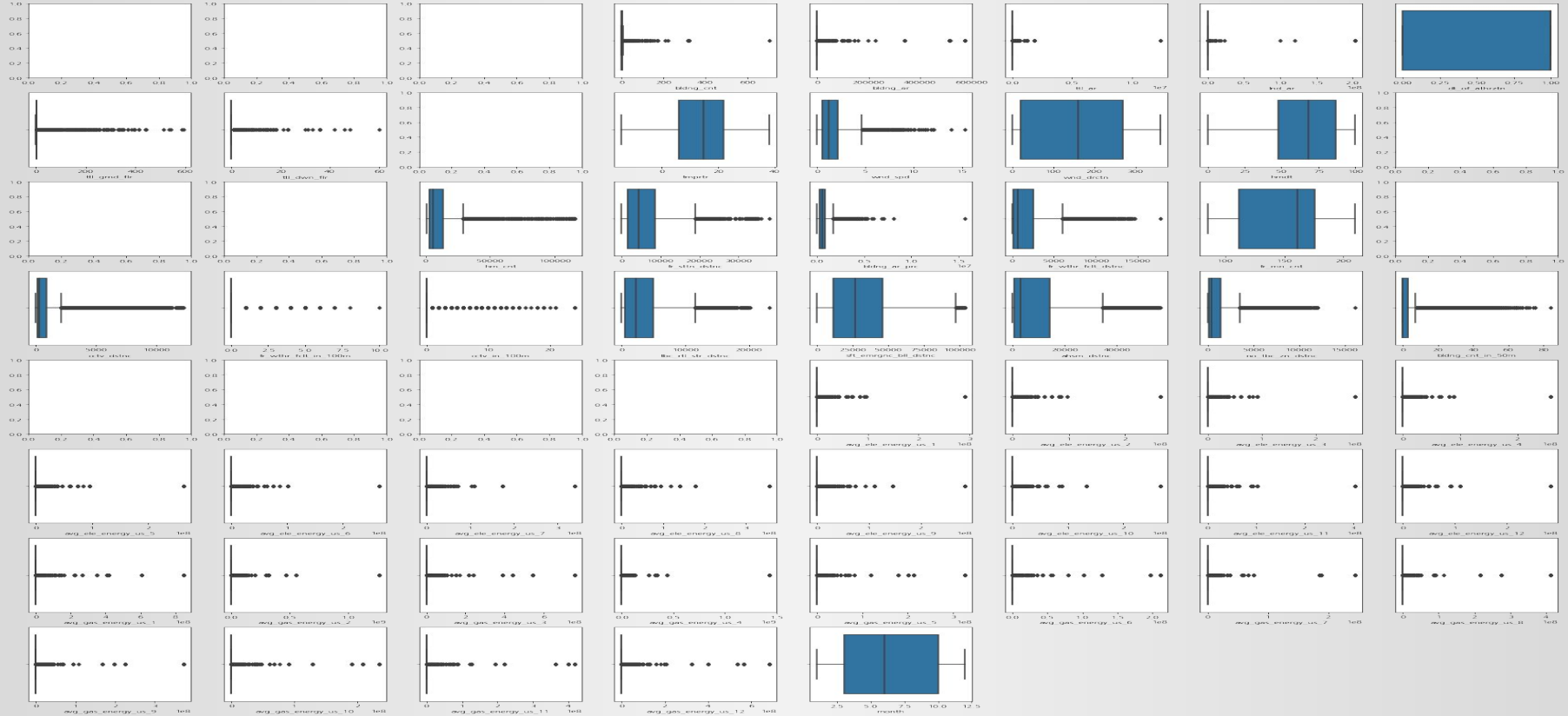
> Binary 변수들 '1','0'으로 변환

화재발생	다중이용시설여부	위험물대상여부	대량위험물제조소등여부	문화재여부
0	0	NaN	NaN	NaN
0	0	NaN	NaN	NaN
0	0	NaN	NaN	NaN
1	0	0.0	N	0.0
0	0	NaN	NaN	NaN
...	...	...	...	...
0	0	NaN	NaN	NaN
0	0	0.0	N	0.0
1	0	0.0	N	0.0
0	0	NaN	NaN	NaN
1	0	NaN	NaN	NaN

> Numerical 변수 처리

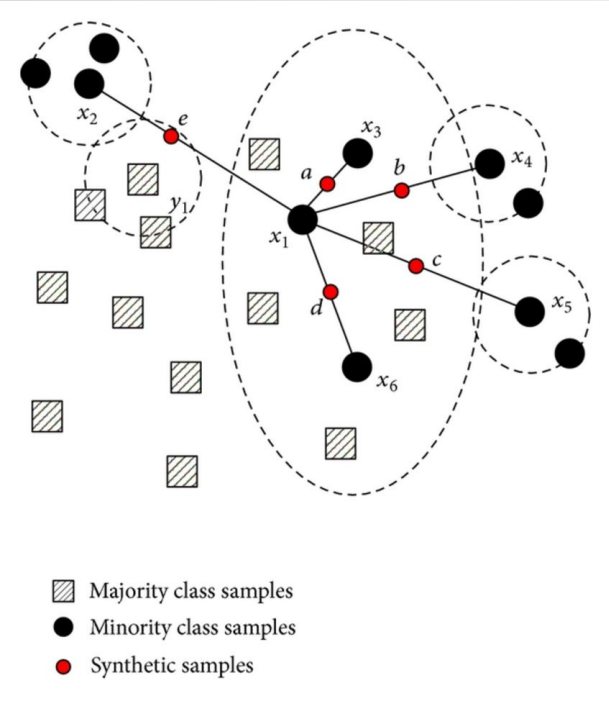
Scaling : 표준화(StandardScaler)

각 데이터 분포


bldng_cnt
bldng_ar
ttl_ar
lnd_ar
rlt_of_albzstn
ttl_grnd_flr
ttl_dwn_flr
tmprtr
wind_spd
wind_drctn
hmdt
hm_cnt
fr_stn_dstnc
bldng_ar_prc
fr_wtr_trll_dstnc
fr_rn_cnt
vctv_dstnc
fr_wtr_trll_rn_100m
vctv_in_100m
lbs_rtl_sh_dstnc
sft_emrgnc_bll_dstnc
shrn_dstnc
rsl_llc_rn_dstnc
bldng_cnt_in_50m
avg_ele_energy_us_1
avg_ele_energy_us_2
avg_ele_energy_us_4
avg_ele_energy_us_5
avg_ele_energy_us_6
avg_ele_energy_us_7
avg_ele_energy_us_8
avg_ele_energy_us_9
avg_ele_energy_us_10
avg_ele_energy_us_11
avg_ele_energy_us_12
avg_gas_energy_us_1
avg_gas_energy_us_2
avg_gas_energy_us_3
avg_gas_energy_us_4
avg_gas_energy_us_5
avg_gas_energy_us_6
avg_gas_energy_us_7
avg_gas_energy_us_8
avg_gas_energy_us_9
avg_gas_energy_us_10
avg_gas_energy_us_11
avg_gas_energy_us_12
month

화재여부 & 각 열 데이터 분포

5) (공통) 데이터 불균형 해결-SMOTE



- 데이터의 불균형을 해결하기위한 메소드 중 하나
- 샘플들 사이의 특성들을 반영한 데이터가 생성되기 때문에 overfitting에 robust한 데이터 생성

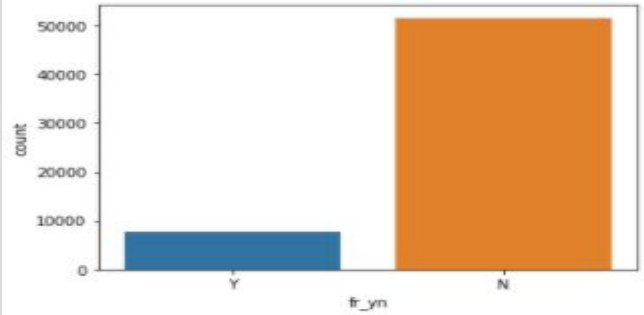
5) (공통) 데이터 불균형 해결-SMOTE

> 화재 여부 데이터를 보면 전체 59199개의 데이터 중 7657개(약 13%)의 실제 화재데이터가 있음

```
import seaborn as sns

sns.countplot(x= 'fr_yn', data = df_train_tmp)
print("Y:" + str((df_train_tmp["fr_yn"]=="Y").sum()))
print("N:" + str((df_train_tmp["fr_yn"]=="N").sum()))
print()
```

```
Y:7657
N:51542
```

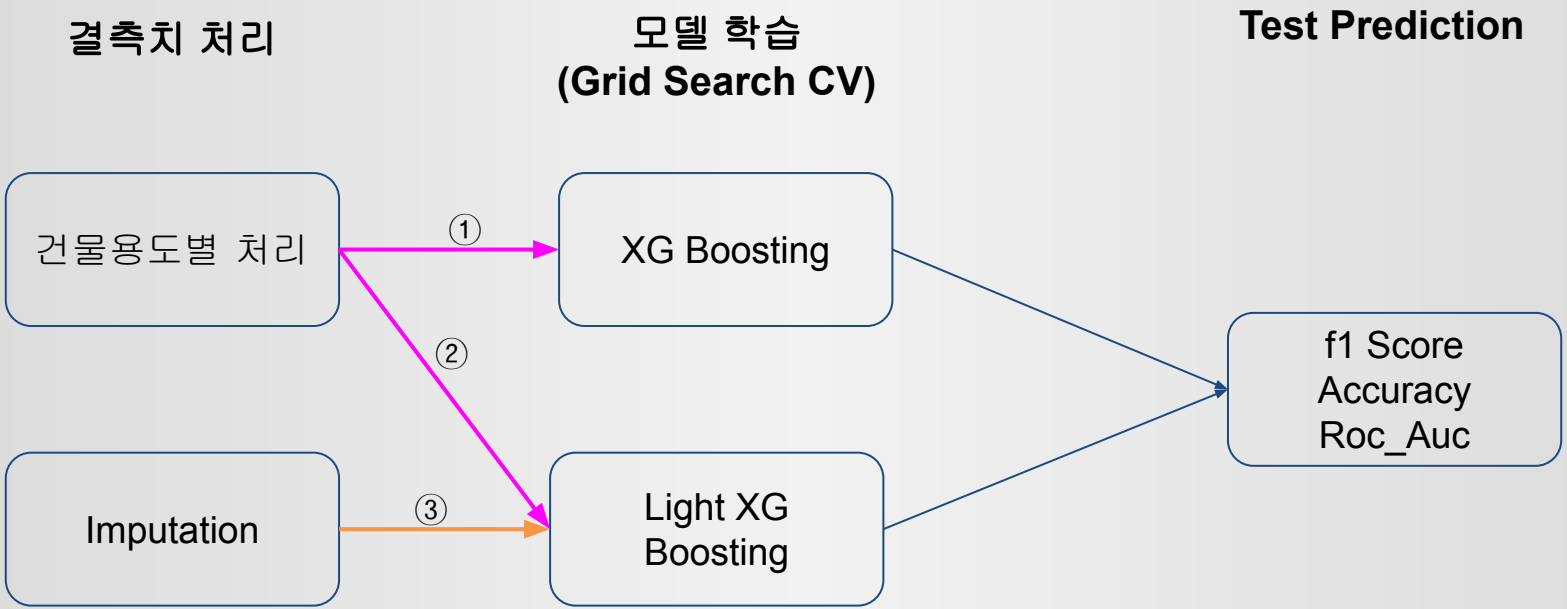


> 불균형을 해소를 위해 SMOTE 알고리즘 적용

```
# 데이터 불균형 해결 - overfitting(SMOTE)
smote = SMOTE(random_state=113)
X_train_smote, y_train_smote = smote.fit_resample(X_train_scd,y_train)
```

03

기존 모델 결과 요약


결측치 처리
모델 학습
(Grid Search CV)
Test Prediction
건물용도별 처리
Imputation
①
②
③
XG Boosting
Light XG
Boosting
f1 Score
Accuracy
Roc_Auc

모델 적용 결과

	XGB (ver1)	LGB (ver 1)	LGB (ver 2)
roc_auc	0.5499	0.6029	0.5975
Accuracy	0.6504	0.6113	0.3715
f1-score	0.2919	0.3587	0.3597

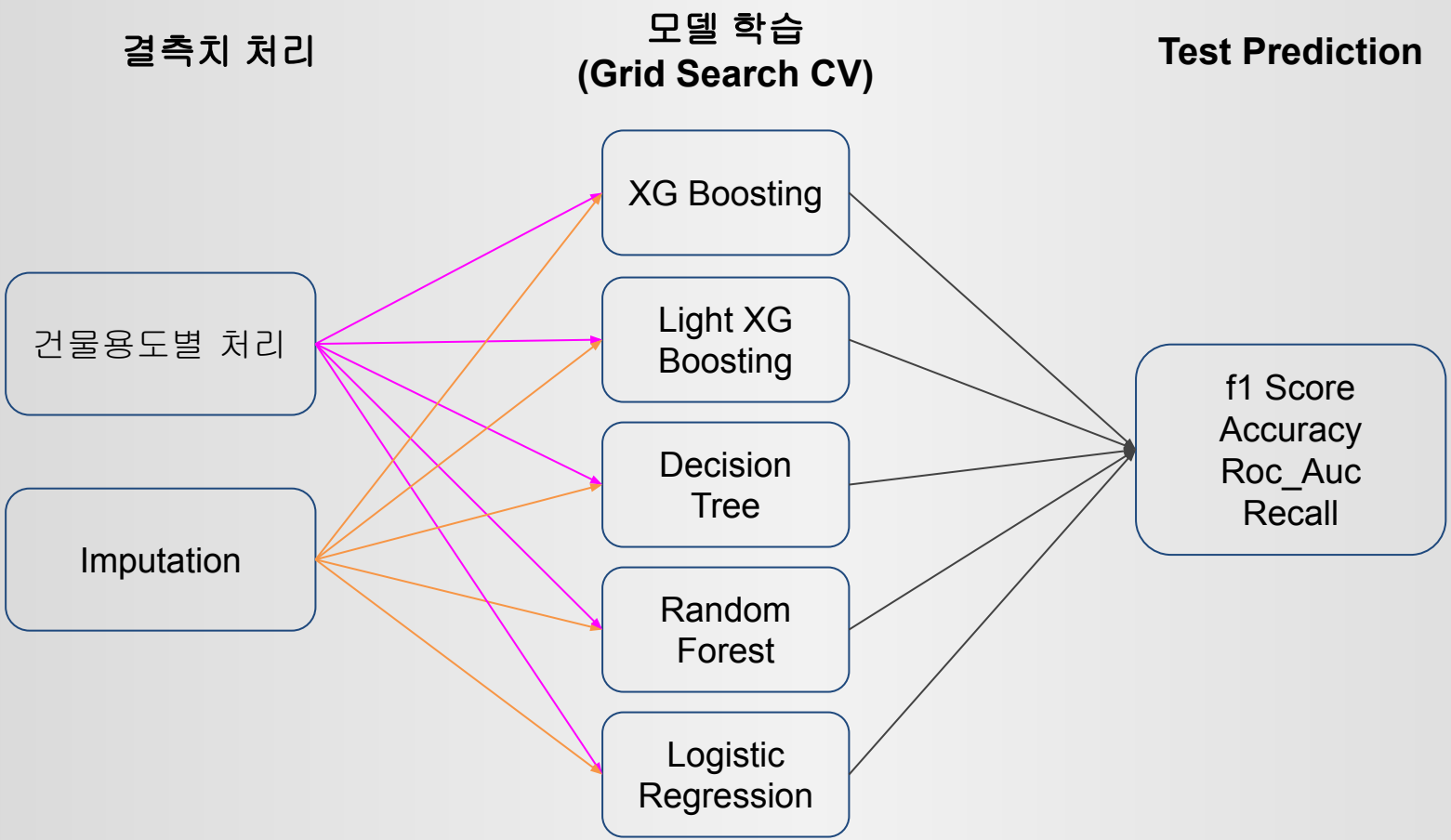
Feedback

- 적은 수의 모델 사용
- f1 스코어 등 테스트 결과가 좋지 않음
- 심층 그리드 서치에 너무 시간을 투자

Feedback 반영

- 모델 확장 : Decision Tree, Random Forest, Logistic Regression
- 김해시 데이터(Validation Data)를 분할, 검증과 테스트 데이터로 사용
  - 훈련 데이터(경상남도)로 모델 학습
  - 검증 데이터(김해시)로 최적 Threshold 탐색
  - 테스트 데이터(김해시)로 최종 모델 평가

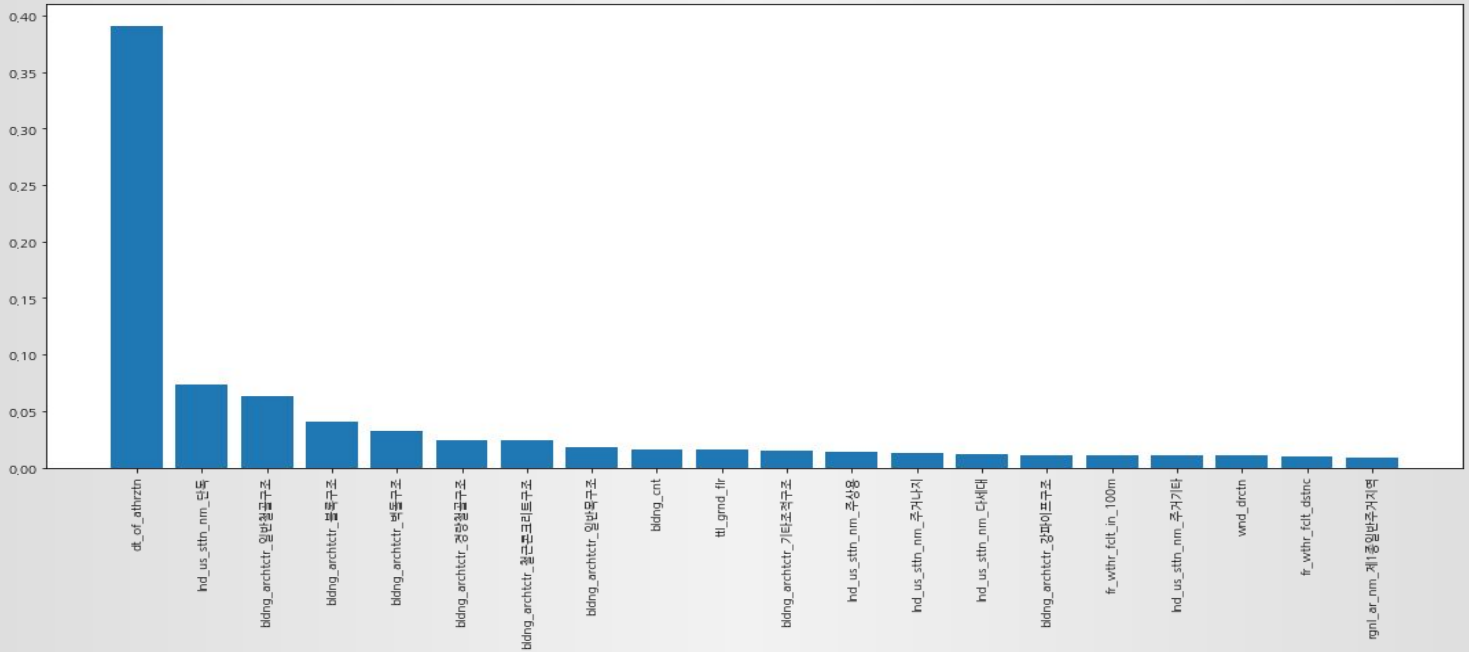
화재발생 예측 모델


결측치 처리
모델 학습
(Grid Search CV)
Test Prediction
건물용도별 처리
Imputation
XG Boosting
Light XG Boosting
Decision Tree
Random Forest
Logistic Regression
f1 Score
Accuracy
Roc_Auc
Recall

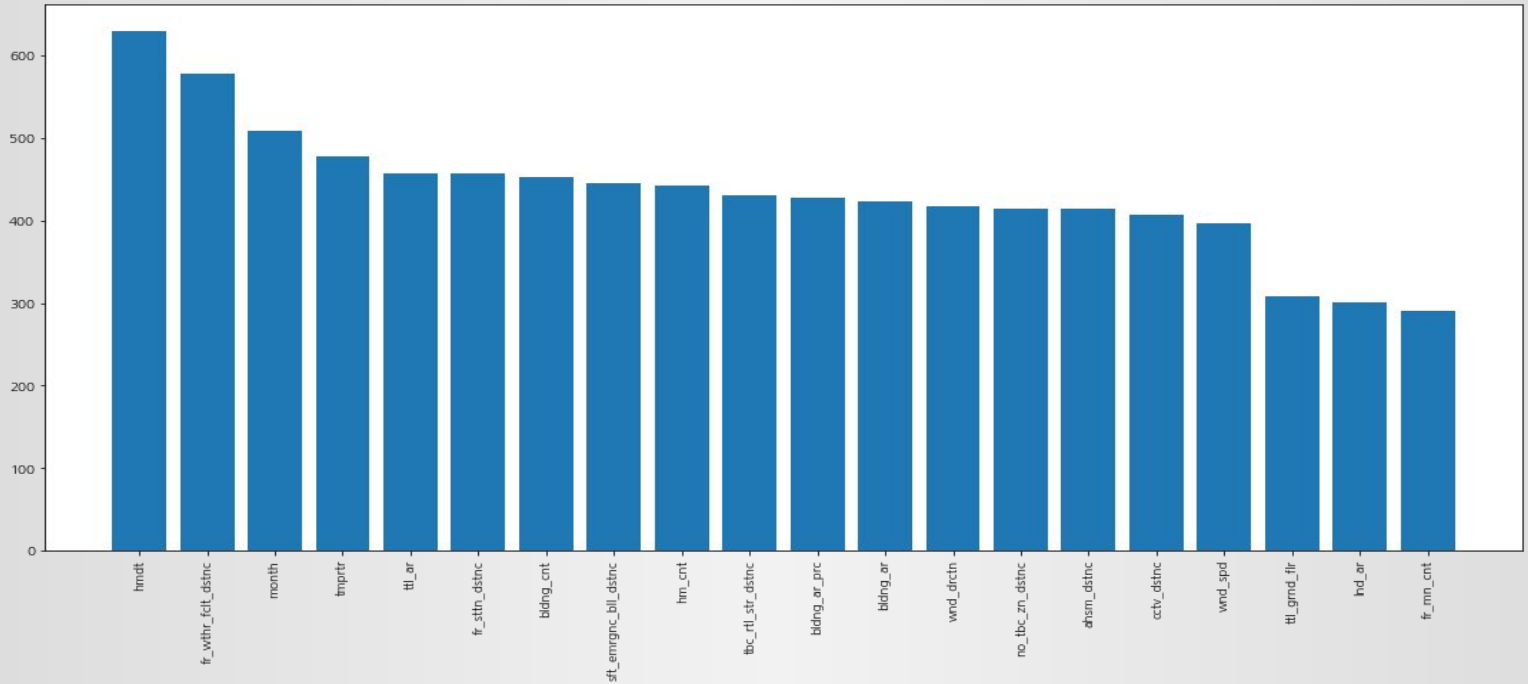
04

피드백 반영 (전처리 version 1)

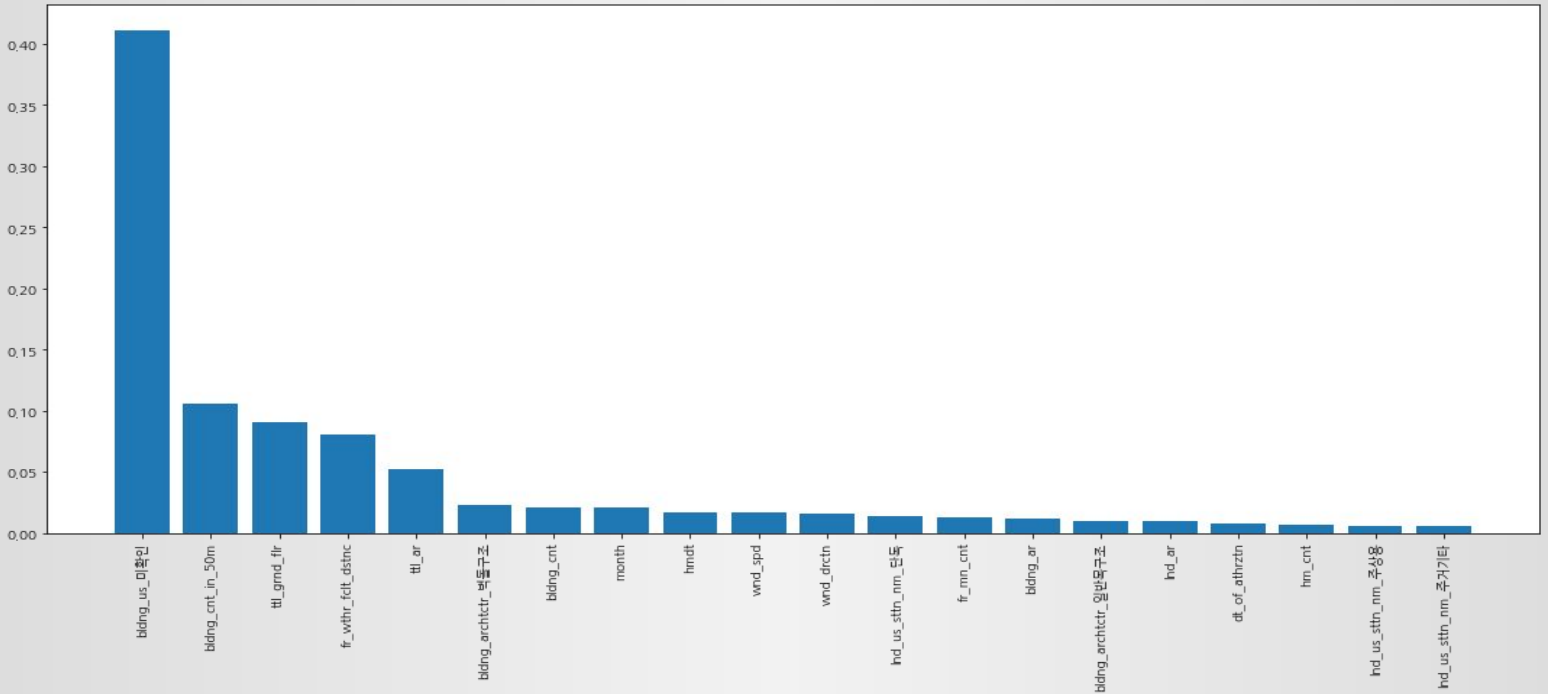
- Best Parameter < objective= binary : logistic, booster = gbtree,
  learning_rate : 1, max_depth : 5 >
- optimal threshold(XGB) : 0.0



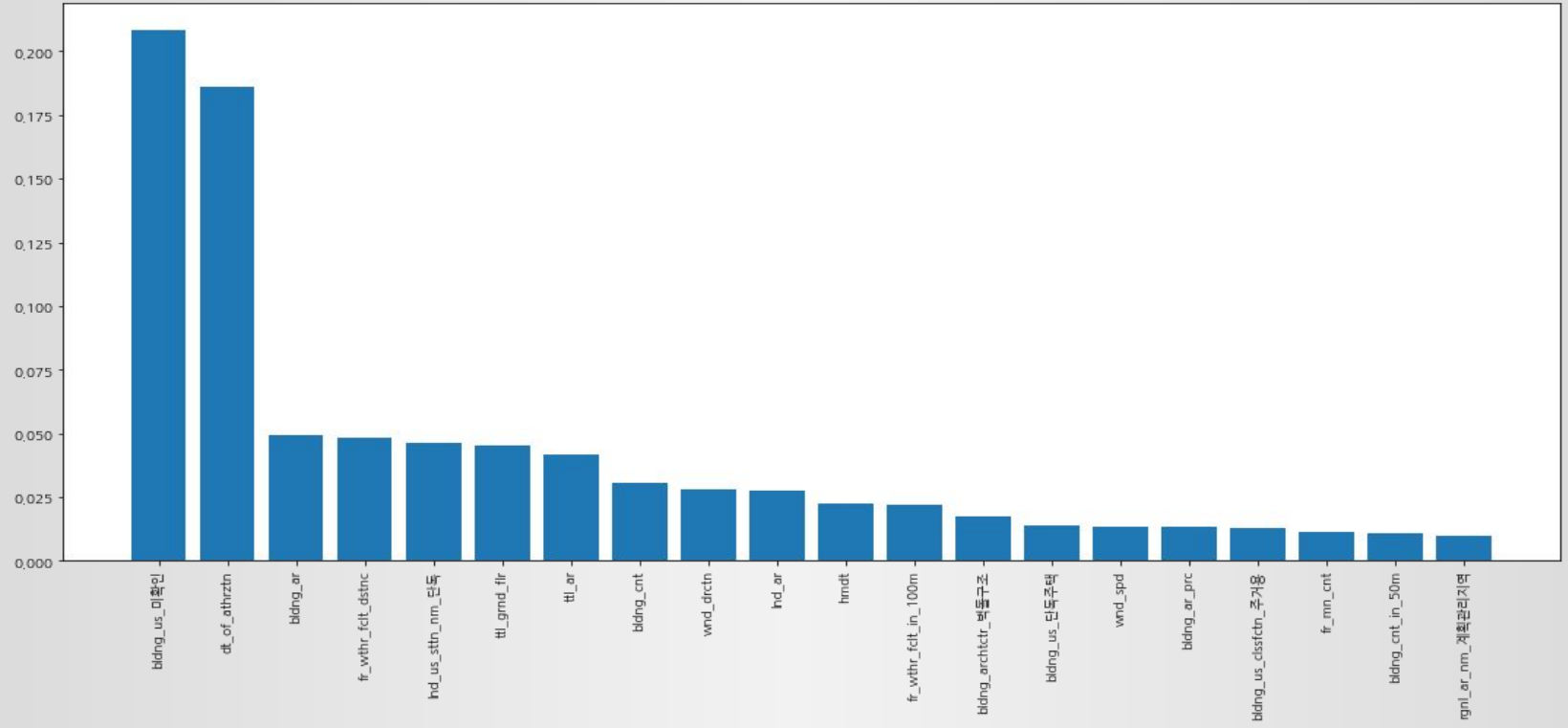
- Best Parameter < learning_rate: 0.01, max_depth : 11, num_leaves : 110 >
- optimal threshold(LGB) : 0.0



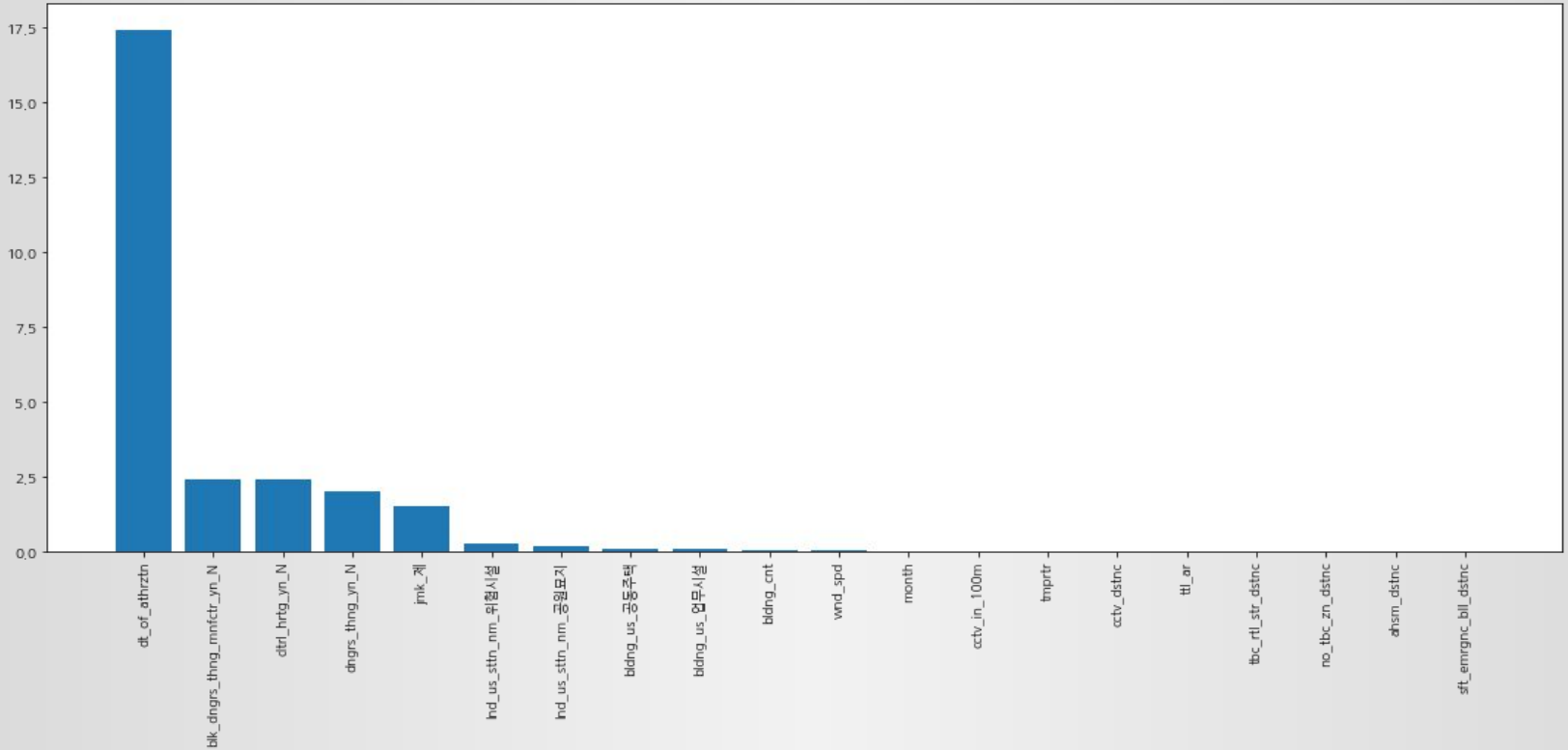
- Best Parameter < criterion : gini, max_depth : 11, max_features : auto, splitter : best >
- optimal threshold(dtc) : 0.0



- Best Parameter < criterion : gini, max_depth : 11, max_features : auto >
- optimal threshold(RF) : 0.32



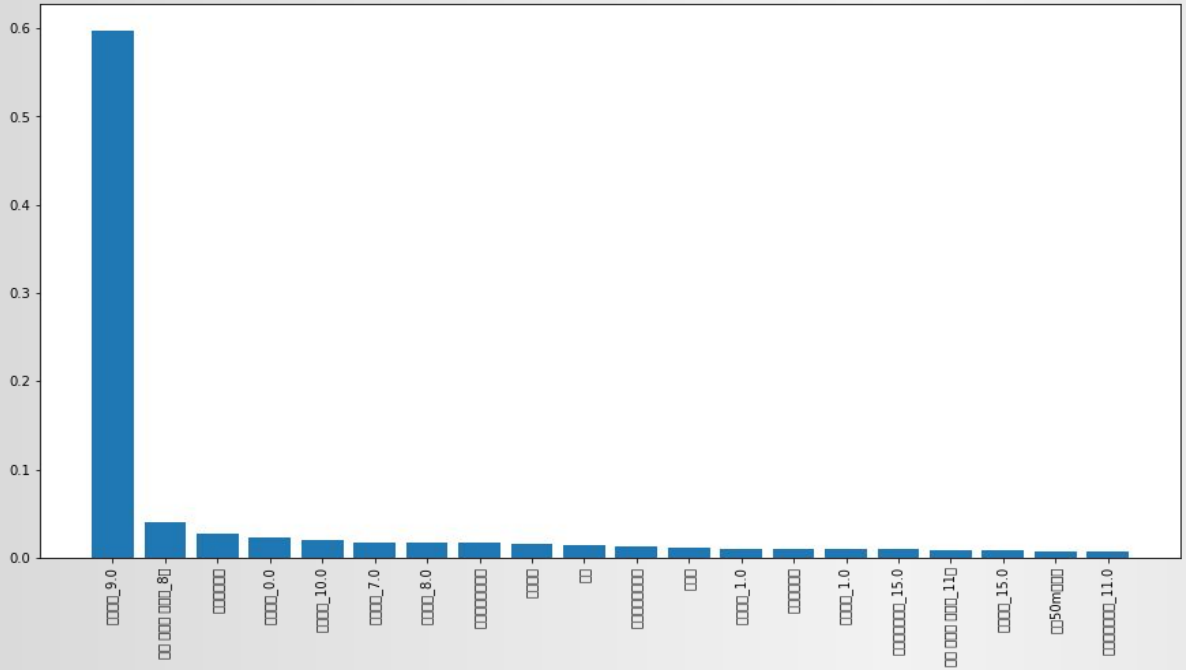
- Best Parameter < penalty : none, solver : newton-cg >
- optimal threshold(lgst) : 0.0



04

피드백 반영 (전처리 version 2)

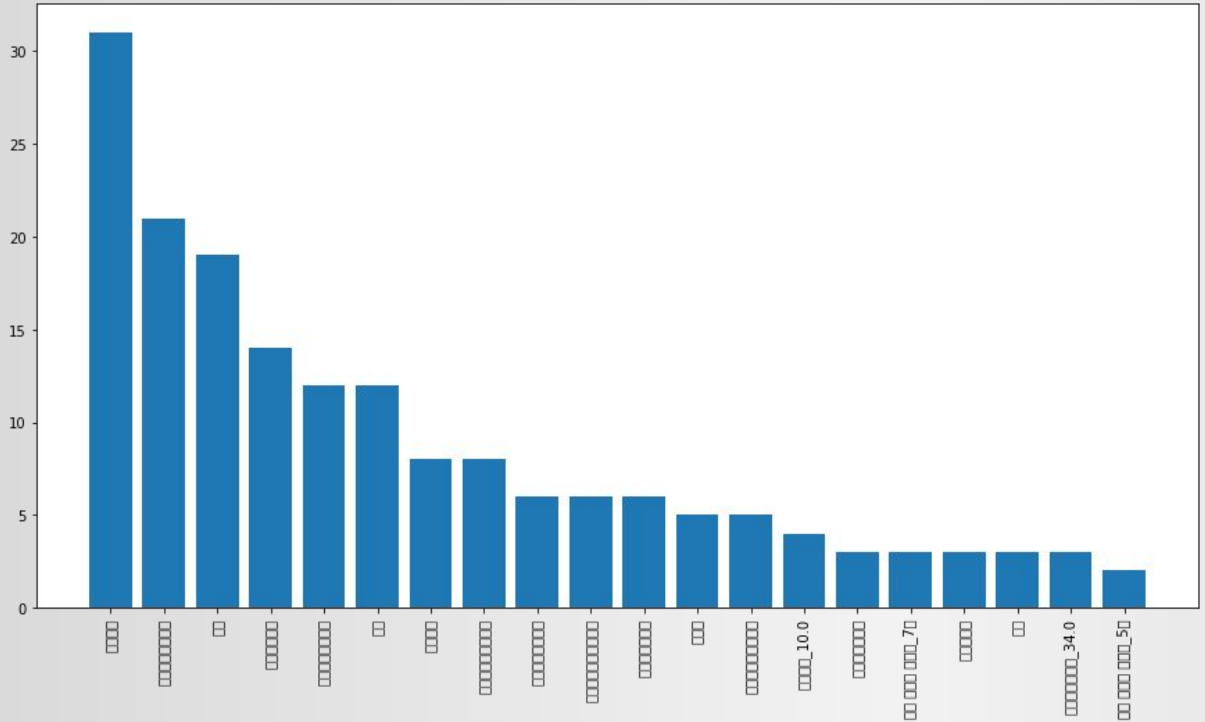
- Best Parameter < objective= binary : logistic, booster = gbtree,
  learning_rate : 1, max_depth : 5 >
- optimal threshold(XGB) : 0.0



<label>

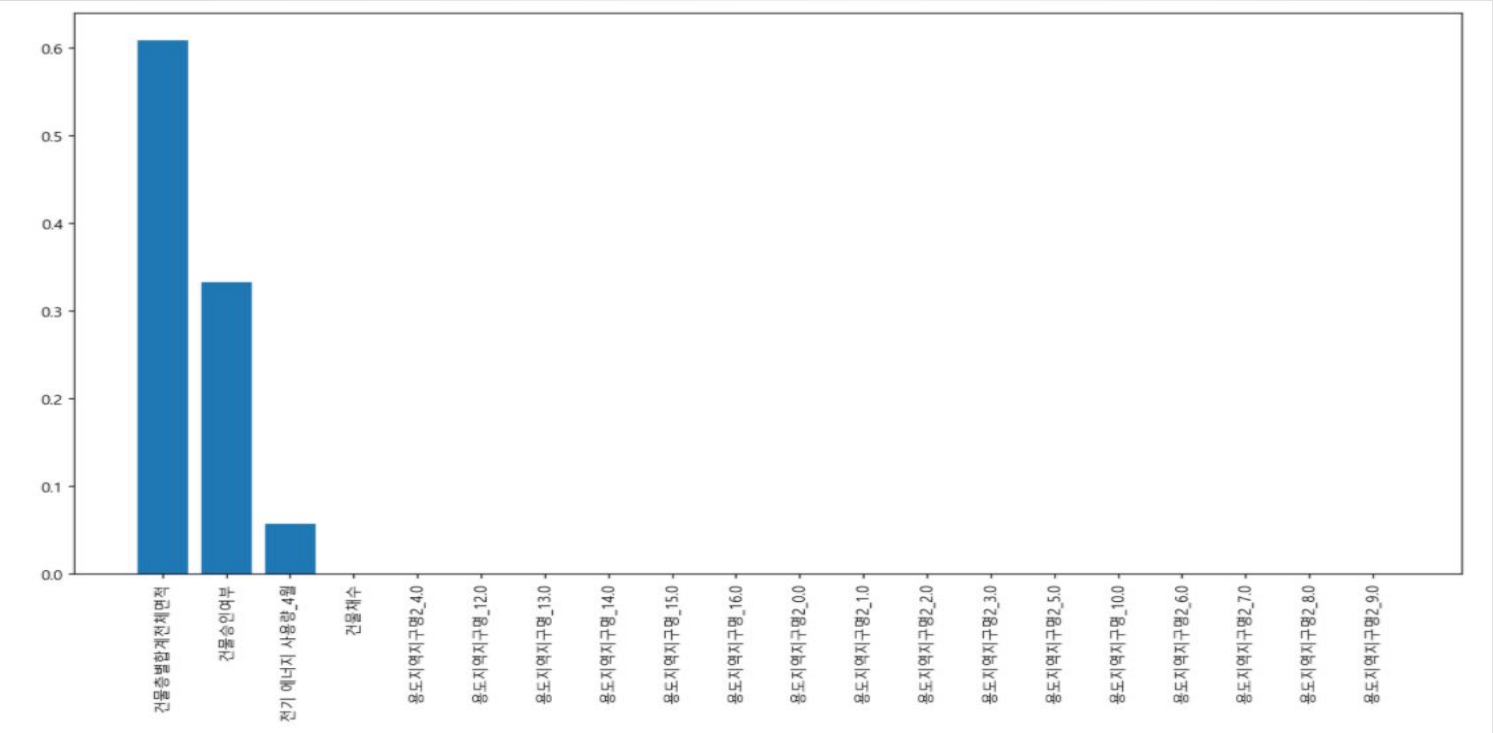
99	건물구조_9.0
45	전기 에너지 사용량_8월
62	건물승인여부
192	사용여부_0.0
100	건물구조_10.0
97	건물구조_7.0
98	건물구조_8.0
4	건물들지상층수합
0	건물채수
8	풍속
28	소방용수시설거리
7	강수량
193	사용여부_1.0
1	건물건축면적
91	건물구조_1.0
129	용도지역지구명_15.0
48	전기 에너지 사용량_11월
105	건물구조_15.0
37	반경50m건물수
158	토지이용상황명_11.0

- Best Parameter < learning_rate: 0.01, max_depth : 11, num_leaves : 110 >
- optimal threshold(LGB) : 0.0

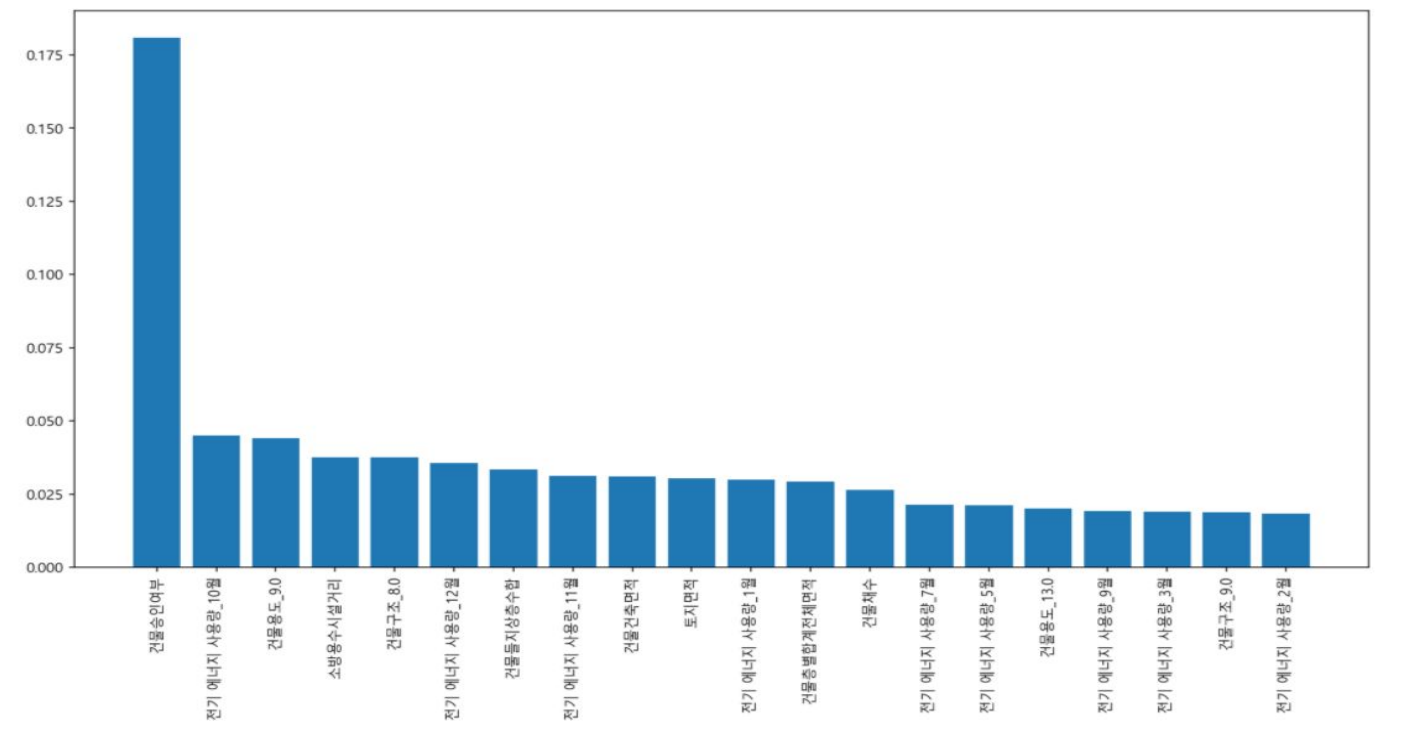


<label>
0 건물채수
4 건물들지상층수합
10 습도
1 건물건축면적
28 소방용수시설거리
8 풍속
3 토지면적
35 자동심장충격기거리
36 금연구역최소거리
2 건물층별합계전체면적
29 관할소방서인원
7 강수량
34 안전비상벨최소거리
100 건물구조_10.0
27 건물면적당가격
44 전기 에너지 사용량_7월
26 소방서거리
9 풍향
181 토지이용상황명_34.0
42 전기 에너지 사용량_5월

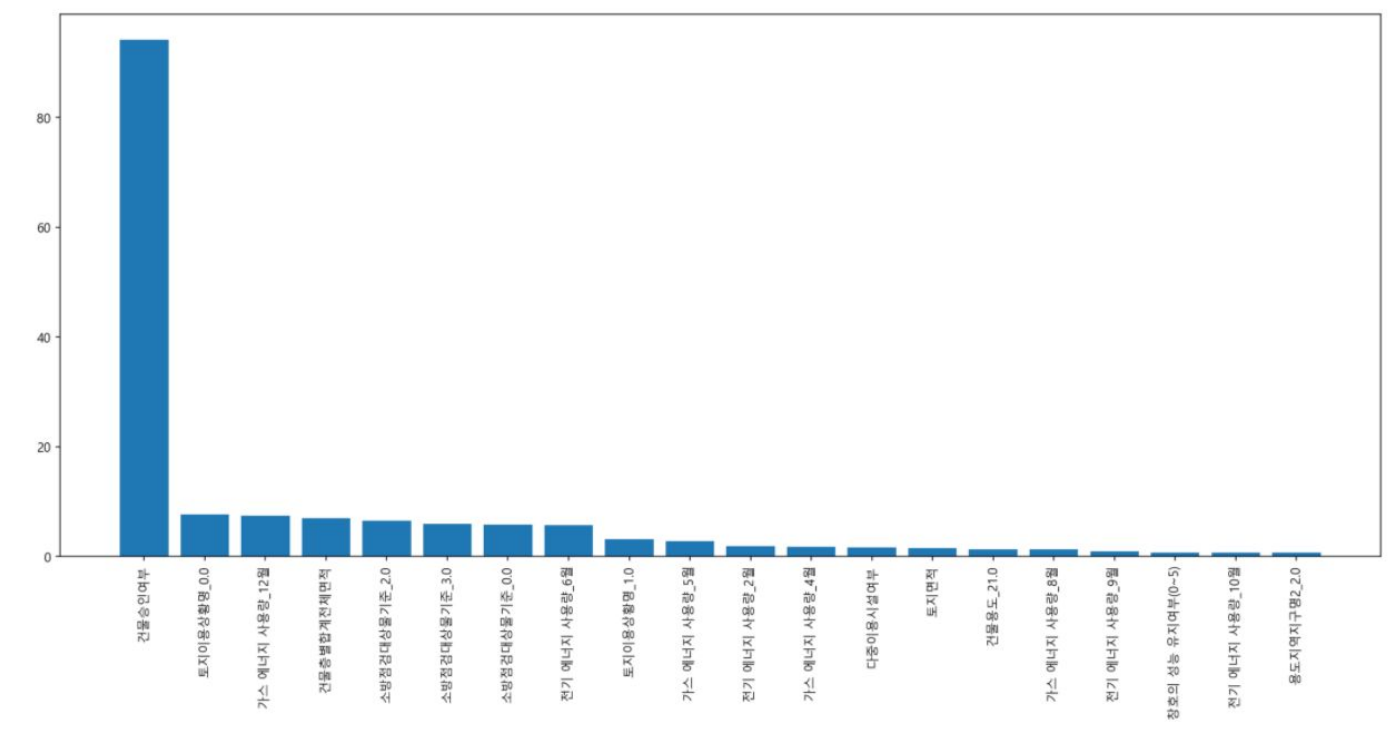
- Best Parameter < criterion : gini, max_depth : 3, max_features : none, splitter : best >
- optimal threshold(dtc) : 0.0



- Best Parameter < criterion : gini, class_weight='balanced', max_features : auto >
- optimal threshold(RF) : 0.59



- Best Parameter < penalty : none, solver :lbfgs >
- optimal threshold(lgst) : 0.0



05

최종 결과

모델 적용 결과

	Decision Tree		Random Forest		Logistic Regression		XGB		LGB	
	ver1	ver2	ver1	ver2	ver1	ver2	ver1	ver2	ver1	ver2
roc_auc	.5	0.7006	.6575	0.6735	.5	0.6711	.5	0.7031	.5	0.6790
accuracy	.1837	0.7970	.6518	0.7091	.1837	0.7567	.1837	0.6961	.1837	0.6300
f1_score	.3104	0.4992	.4130	0.4393	.3104	0.4483	.3104	0.4647	.3104	0.4303
recall	1.0	0.5478	.6667	0.6173	1.0	0.5353	1.0	0.7142	1.0	0.7566

THE END

Thank you