오픈소스SW
설계

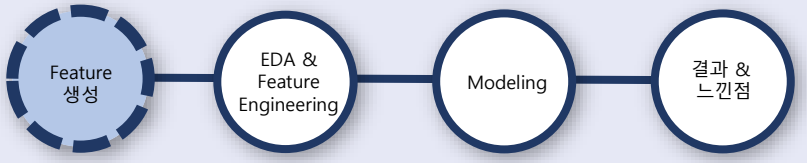
단국대 소중 데이터 분석 공모전

팀명 : Old Pine

응용통계학과 32152339 송준영(바디빌더 스펀지송)
응용통계학과 32161384 노승찬(Rrohchan)

오픈소스SW

CONTENTS



A 조합

```python
#조합으로 연산 피쳐 생성
fea2=['u','g','r','i','z','redshift','dered_u','dered_g',
      'dered_r','dered_i','dered_z','nObserve','nDetect']
for c1,c2 in tqdm(itertools.combinations(fea2,2)):
    dif_col=f'diff_{c1}_{c2}'
    div_col=f'div_{c1}_{c2}'
    sum_col=f'sum_{c1}_{c2}'
    mul_col=f'mul_{c1}_{c2}'
    df[dif_col]=df[c1]-df[c2]
    df[div_col]=df[c1]/df[c2]
    df[sum_col]=df[c1]+df[c2]
    df[mul_col]=df[c1]*df[c2]
```

- **Feature set ver.1**
- **변수들의 여러 조합을 사용하여 변수 생성**
- **사칙연산 사용**

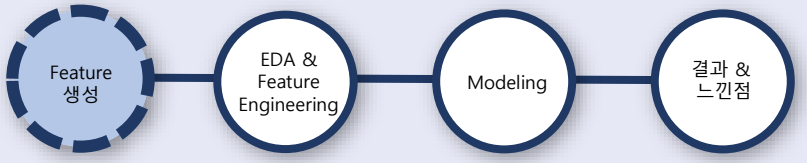
기존 변수들을 사용하여 여러 변수 생성

B 통계치

```python
#카테고리별 통계치 변수 생성
#max-min
df['max-min'] = df[all_fea].max(axis=1)-df[all_fea].min(axis=1)
df['max-min_ugriz'] = df[ugriz].max(axis=1)-df[ugriz].min(axis=1)
df['max-min_dered'] = df[dered].max(axis=1)-df[dered].min(axis=1)
#std
df['std'] = df[all_fea].std(axis=1)
df['std_ugriz'] = df[ugriz].std(axis=1)
df['std_dered'] = df[dered].std(axis=1)
#파장별 합
df['sum'] = df[all_fea].sum(axis=1)
df['sum_ugriz'] = df[ugriz].sum(axis=1)
df['sum_dered'] = df[dered].sum(axis=1)
#파장별 최대값
df['max'] = df[all_fea].max(axis=1)
df['max_ugriz'] = df[ugriz].max(axis=1)
df['max_dered'] = df[dered].max(axis=1)
#파장별 최소값
df['min'] = df[all_fea].min(axis=1)
df['min_ugriz'] = df[ugriz].min(axis=1)
df['min_dered'] = df[dered].min(axis=1)
#파장별 max-max,min=min,sum-sum
df['max-max']=df[ugriz].max(axis=1)-df[dered].max(axis=1)
df['min-min']=df[ugriz].min(axis=1)-df[dered].min(axis=1)
df['sum-sum']=df[ugriz].sum(axis=1)-df[dered].sum(axis=1)
#왜도,첨도 구하기
df['skew']=skew(df[ugriz],axis=1)
df['kurtosis']=kurtosis(df[ugriz],axis=1)
df['dered_skew']=skew(df[dered],axis=1)
df['dered_kurtosis']=kurtosis(df[dered],axis=1)
df['airmass_skew']=skew(df[airmass],axis=1)
df['airmass_kurtosis']=kurtosis(df[airmass],axis=1)
```

- **Feature set ver.1**
- **통계치를 이용하여 변수 생성**
- **행별 또는 변수별로 적용**

최대값, 최소값, 표준편차, 왜도, 첨도



C 도메인

```
# 도메인 정보를 바탕으로 변수 추가
# 출처 : http://classic.sdss.org/dr6/algorithms/sdssUBVRITransform.html
df['asinh_mu'] = -2.5/np.log(10)*(np.arcsinh(df.u/24.63/(2.8e-10))-22.689378693319245)
df['asinh_mg'] = -2.5/np.log(10)*(np.arcsinh(df.g/25.11/(1.8e-10))-23.131211445598282)
df['asinh_mr'] = -2.5/np.log(10)*(np.arcsinh(df.r/24.80/(2.4e-10))-22.843529373146502)
df['asinh_mi'] = -2.5/np.log(10)*(np.arcsinh(df.i/24.36/(3.6e-10))-22.43806426503834)
df['asinh_mz'] = -2.5/np.log(10)*(np.arcsinh(df.z/22.83/(1.48e-09))-21.024370929730330)

df['Q_U_B']=0.75*(df.u-df.g)-0.81
df['Q_B_V']=0.62*(df.g-df.r)+0.15
df['Q_V_R']=0.38*(df.r-df.i)-0.2
df['Q_Rc_IC']=0.72*(df.r-df.i)-0.27
df['Q_B']=df.g+0.17*(df.u-df.g)+0.11
df['Q_V']=df.g-0.52*(df.g-df.r)-0.03
df['S_U_B']=0.77*(df.u-df.g)-0.88
df['S_B_V']=0.90*(df.g-df.r)+0.21
df['S_V_R']=0.96*(df.r-df.i)+ 0.21
df['S_Rc_IC']=1.02*(df.r-df.i)+0.21
df['S_B']=df.g+0.33*(df.g-df.r)+0.20
df['S_V']=df.g -0.58*(df.g-df.r)- 0.01
df['I-color'] = (-0.436*df['u']) + (1.129*df['g']) - (0.119*df['r']) - (0.574*df['i']) + (0.1984)
df['s-color'] = (-0.249*df['u']) + (0.794*df['g']) - (0.555*df['r']) + (0.234)

df['domain1'] = 0.7*(df.g-df.r) + 1.2*((df.r-df.i) - 0.177)
df['domain2'] = (df.r-df.i) - (df.g-df.r)/4 - 0.177
df['domain3'] = 0.449 - (df.g-df.r)/6
df['domain4'] = 1.296 + 0.25*(df.r-df.i)
df['domain5'] =  (df.r-df.i) - (df.g-df.r)/4 - 0.18
```

- Feature set ver.2
- 도메인 지식을 활용하여 추가적으로 변수 생성
- sdss 홈페이지 참조

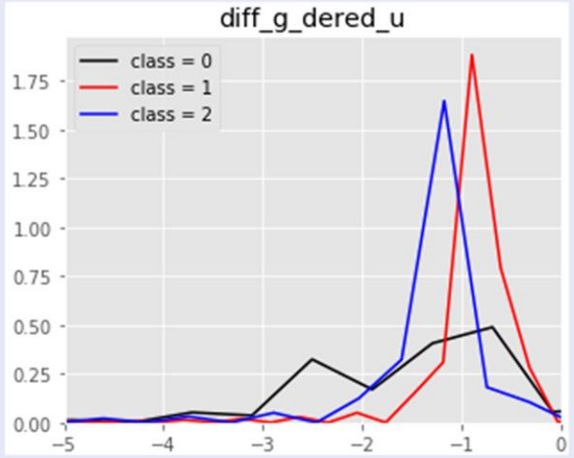
https://www.sdss.org/dr16/algorithms/segue_target_selection/#Legacy

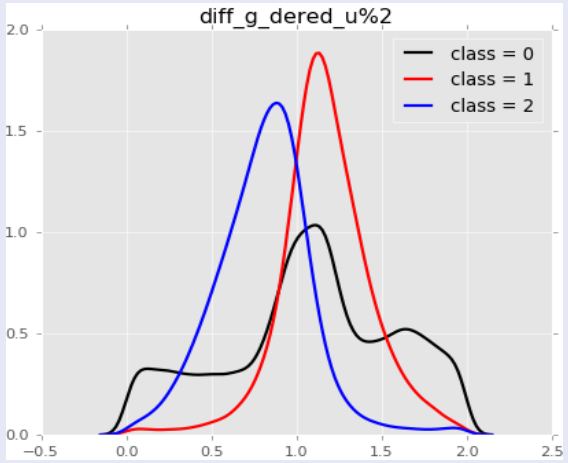
D redshift

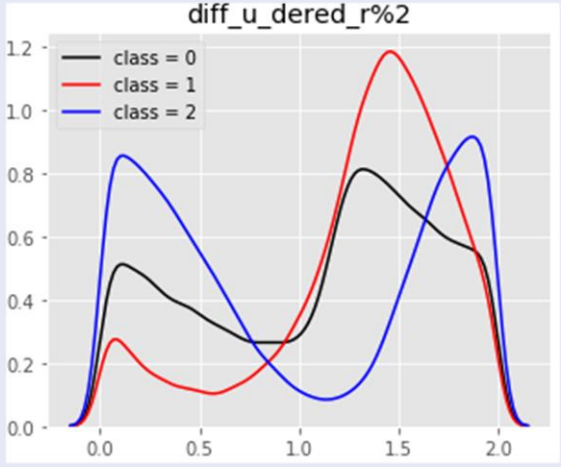
```
df['redshift_u_g'] = df.redshift-df.u-df.g
df['redshift_u_r'] = df.redshift-df.u-df.r
df['redshift_g_r'] = df.redshift-df.g-df.r
df['redshift_4_log_u_2'] = df.redshift**4+np.log(df.u**2)

for c in ugriz:
    div_col=f'div_redshift_2_{c}'
    df[div_col]=df.redshift**2/df[c]
```

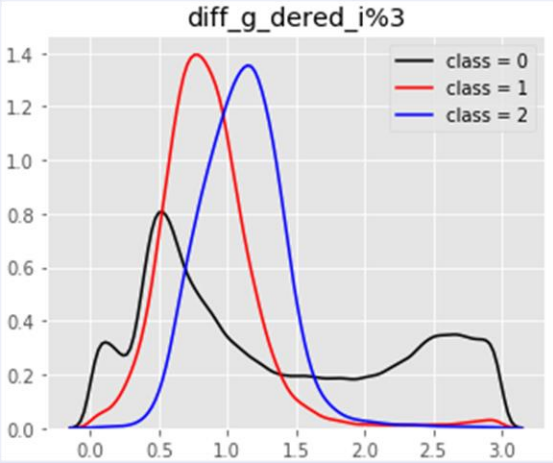
- Feature set ver.2
- RF feature importance 1위 redshift 관련 연산 함수 생성

A 분류를 잘 하는 변수

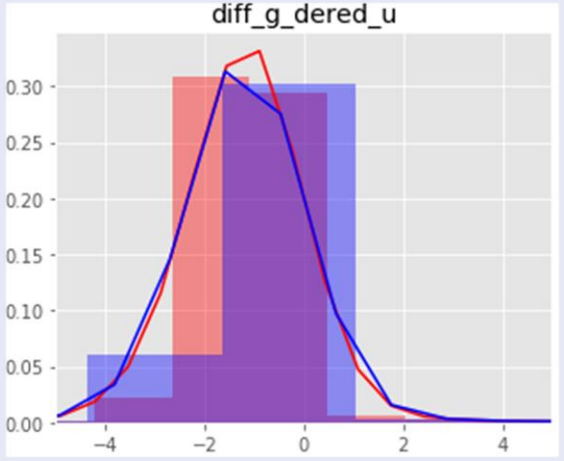


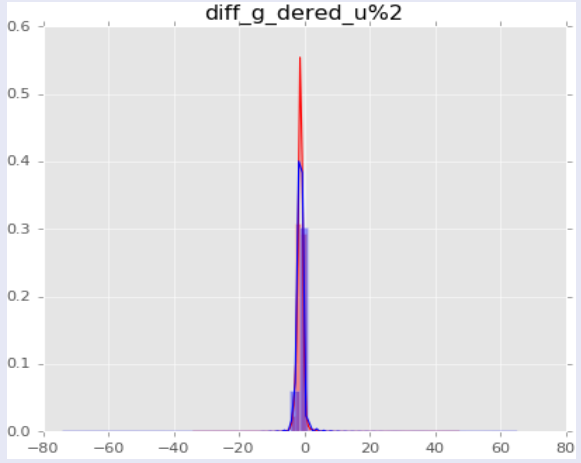


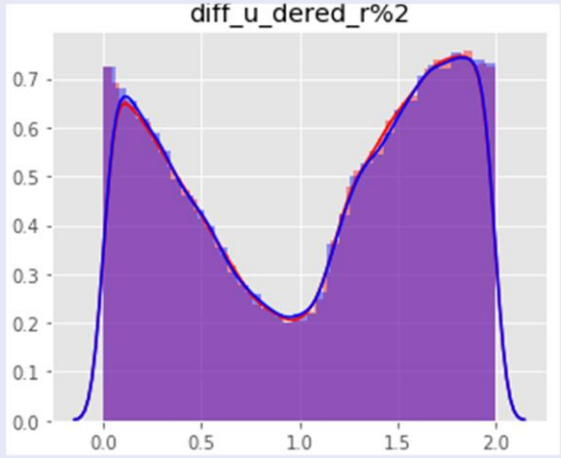


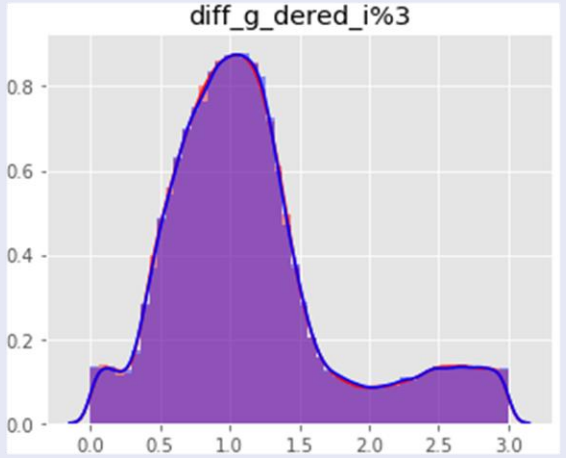


B TST 데이터와 분포 비교









A 차원 축소

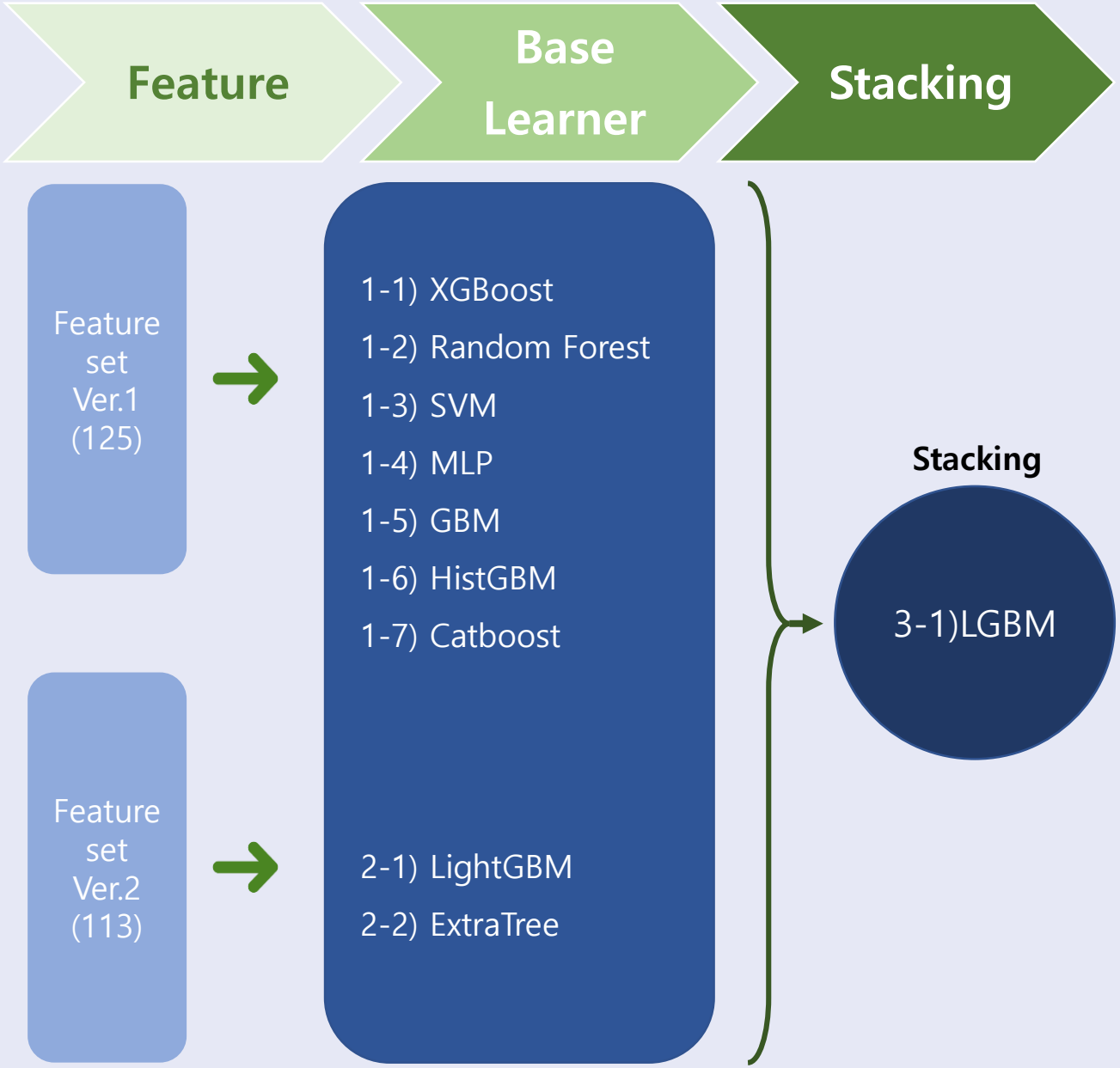
```
model=lgbm_wrapper.fit(ftr,target,eval_metric='multi_error',verbose=True)
```

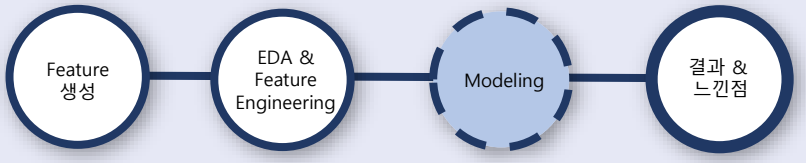
```
perm = PermutationImportance(model, scoring = "accuracy").fit(ftr, target)
eli5.show_weights(perm, top = ftr.shape[1], feature_names = ftr.columns.tolist())
```

Weight	Feature
0.0083 ± 0.0004	diff_dered_u_dered_g
0.0074 ± 0.0004	diff_dered_r_dered_i
0.0059 ± 0.0003	div_dered_r_dered_i
0.0044 ± 0.0003	diff_r_i
0.0042 ± 0.0003	div_dered_u_dered_r
0.0037 ± 0.0003	diff_dered_g_dered_r
0.0030 ± 0.0003	s-color
0.0028 ± 0.0003	div_dered_g_dered_r
0.0027 ± 0.0001	div_redshift_nDetect
0.0022 ± 0.0001	div_i_dered_r
0.0021 ± 0.0002	l-color
0.0020 ± 0.0002	diff_dered_i_dered_z
0.0020 ± 0.0002	diff_g_dered_u
0.0018 ± 0.0002	diff_g_r
0.0018 ± 0.0002	div_r_i
0.0016 ± 0.0002	div_g_r
0.0016 ± 0.0001	diff_dered_u_dered_r
0.0016 ± 0.0001	div_r_dered_g
0.0015 ± 0.0002	diff_i_dered_r
0.0015 ± 0.0003	div_r_dered_u
0.0015 ± 0.0002	diff_u_g
0.0013 ± 0.0001	div_u_r
0.0012 ± 0.0002	mul_r_redshift
0.0012 ± 0.0002	mul_i_redshift
0.0012 ± 0.0001	div_dered_u_dered_g
0.0012 ± 0.0001	div_dered_i_dered_z
0.0012 ± 0.0001	redshift%14
0.0011 ± 0.0002	diff_r_dered_g
0.0011 ± 0.0001	diff_i_z

-0.0000 ± 0.0002	div_redshift_nObserve
-0.0000 ± 0.0001	dered_r
-0.0000 ± 0.0001	dered_z
-0.0000 ± 0.0001	dered_g
-0.0000 ± 0.0002	div_dered_g_dered_z
-0.0000 ± 0.0000	sum_dered_i_nDetect
-0.0000 ± 0.0000	mul_dered_g_dered_z
-0.0000 ± 0.0000	sum_z_dered_i
-0.0001 ± 0.0001	diff_r_redshift
-0.0001 ± 0.0001	sum_u_redshift
-0.0001 ± 0.0000	sum_dered_r_nDetect
-0.0001 ± 0.0000	mul_g_dered_z
-0.0001 ± 0.0000	sum_dered_r_dered_z
-0.0001 ± 0.0000	div_r_nObserve
-0.0001 ± 0.0001	sum_i_redshift
-0.0001 ± 0.0001	diff_dered_g_dered_z
-0.0001 ± 0.0001	sum_dered_g_dered_r
-0.0001 ± 0.0000	sum_r_dered_z
-0.0001 ± 0.0001	diff_u_redshift
-0.0001 ± 0.0000	mul_u_dered_z
-0.0001 ± 0.0000	mul_dered_g_dered_r
-0.0001 ± 0.0000	mul_i_dered_r
-0.0001 ± 0.0001	div_i_dered_g
-0.0001 ± 0.0001	sum_r_redshift
-0.0001 ± 0.0001	div_u_dered_z
-0.0002 ± 0.0000	mul_r_z
-0.0002 ± 0.0001	div_g_z
-0.0002 ± 0.0000	diff_u_dered_z
-0.0002 ± 0.0001	mul_r_dered_g
-0.0004 ± 0.0001	sum_redshift_dered_z
-0.0006 ± 0.0001	u
-0.0009 ± 0.0002	div_redshift_dered_u

- 생성한 365개의 변수(feature ver.1)를 LightGBM 기반 Permutation importance(eli5)를 사용하여 125개로 축소
  - 20개씩 변수를 줄여가면 CV값이 가장 높은 변수 개수로 선정
- 이후 MLP Classifier와 SVM 모델을 사용할때는 상관계수를 기준으로 42개, 31개의 변수로 축소시킴
  - 상관계수가 0.9 이상인 변수쌍 중에서 한 개를 남기고 나머지 제거
- 125개의 변수를 더 줄이기 위해 PCA와 LDA등의 차원 축소 기법을 시도했으나 lightGBM과 Random Forest의 cv accuracy 및 stacking 후의 test accuracy 모두 성능 저하





1-1) XGBoost
booster를 'dart'로 설정했을 때 성능이 더 올라감. 개별 성능 상위.

1-2) Random Forest
Base learner 중 93.7정도로 성능이 가장 우수.

1-3) SVM
단일로는 성능이 좋지않아 BaggingClassifier와 함께 사용. 개별성능은 92 정도이나 stacking 시 성능 향상에 큰 도움이 되었음.

1-4) MLP
Tree 계열 모델이 많아 추가. 개별 성능은 93.3 정도이나 stacking 시 성능 향상에 큰 도움이 되었음.

1-5) GBM
모든 파라미터를 default로 설정.

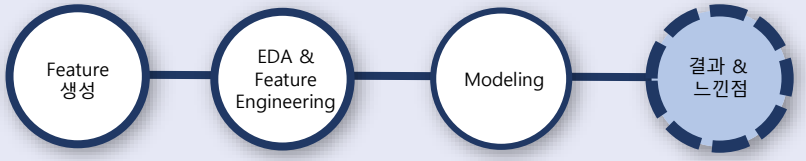
1-6) HistGBM
Max_iter를 늘렸을 때 개별 성능은 좋아졌으나 stacking 시 저하

1-7) Catboost
개별 성능은 gpu를 사용한 모델이 더 우수하나, stacking 시 cpu를 사용한 모델이 성능 향상에 더 도움됨.

2-1) LightGBM
파라미터 설정에 가장 많은 공을 들인 모델, 개별 성능이 RF와 함께 우수

2-2)ExtraTree
Random Forest와 비슷한 성능을 보이나 Stacking 시 성능 향상에 큰 도움

3-1) LightGBM
Stacking 할 모델 중 RF, XGBoost, LightGBM 중 가장 성능이 좋았던 LightGBM 선택. 최종 5fold cv accuracy : 0.9382



[결과]

- Modeling part보다 Feature engineering part가 성능 향상에 보다 직접적으로 영향
- stacking을 사용한 이후 CV값이 안정적으로 0.938을 넘김
- Base learner들의 성능이 가장 우수했던 Feature_ver.1과 ver.2 만을 사용한 stacking model이 가장 성능이 좋았음
- 최종 private score 약 0.93765로 1등 기록

[느낀점]

- Python을 처음 다루어 대부분 hard coding으로 이루어져 수행시간이 오래 걸림.
- EDA를 통해 얻은 변수가 성능 향상에 도움 되지 않는 것이 더 많았음
- 여러방법의 스태킹(Feature별, Model별)을 해보았으나 과적합의 이유를 찾지 못함
- 모델을 조금 더 추가하려 했으나 Memory Error와 같은 hardware 문제로 사용 제한
- 개별 base learner의 성능이 우수하면 반드시 stacking model 성능이 향상되는 것이 아닌, 모델이나 피쳐가 얼마나 다양한가(독립적인가)가 중요하다는 것을 느낌

감사합니다

Old Pine