2022-2 KUBIG CONTEST
머신러닝 분반 3팀

월간 DACON

유권자 심리 성향 예측 프로젝트

16기 **김상옥** | 식품자원경제학과 2021140655
16기 **민윤기** | 보건환경융합과학부 2018250287
16기 **신인섭** | 바이오의공학부 2018250066
16기 **윤지현** | 바이오의공학부 2020250046

유권자 심리 성향 예측 프로젝트 2022-2 KUBIG CONTEST 머신러닝 3팀

CONTENTS

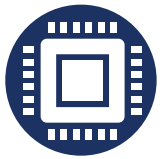
발표 순서

01 Raw Data EDA

- Feature 구성
- Target: Voted
- Outlier
- 무응답 결측치

02 Data Preprocessing

- Log Transformation
- 무응답 대치
- PCA



03 Modeling

세 가지 Case:

1. 전처리 X, AutoML
2. 전처리 & 직접 모델링
3. 전처리 & AutoML

04 Result

- Test set 예측 제출
- 의의와 한계

01

Raw Data EDA

유권자 심리 성향 예측 프로젝트 2022-2 KUBIG CONTEST 머신러닝 3팀

01 Raw Data EDA

Feature 구성

```
[ ]  train.head()
```

index	QaA	QaE	QbA	QbE	QcA	QcE	QdA	QdE	QeA	QeE	...
0	3.0	363	4.0	1370	5.0	997	1.0	1024	2.0	1577	...
1	5.0	647	5.0	1313	3.0	3387	5.0	2969	1.0	4320	...
2	4.0	1623	1.0	1480	1.0	1021	4.0	3374	5.0	1333	...
3	3.0	504	3.0	2311	4.0	992	3.0	3245	1.0	357	...
4	1.0	927	1.0	707	5.0	556	2.0	1062	1.0	1014	...

5 rows × 77 columns

Feature 76개, Target 1개(voted)

- Train Set 관측치 45532개
- Test Set 관측치 11383개

설문 응답 변수

- Q□A : 질문에 대한 응답 (1 to 5)
- Q□E : 질문에 응답하기까지 걸린 시간

(□ = a ~ t까지)

응답자 인적 사항

age_group(연령대), education(교육 수준 0 to 4), engnat(영어 모국어 여부), familysize(형제자매 수), gender(성별), hand(필기하는 손), married(혼인 상태), race(인종), religion(종교), urban(유년기 거주 구역)

응답자 자기 인식

"I see myself as:" □□□□, 형용사 10개 세트

어휘 이해 여부

- wr□(01~13) : 실존 단어 정의를 안다고 응답(0 or 1)
- wf□(01~03) : 허구 단어의 정의를 안다고 응답(0 or 1)

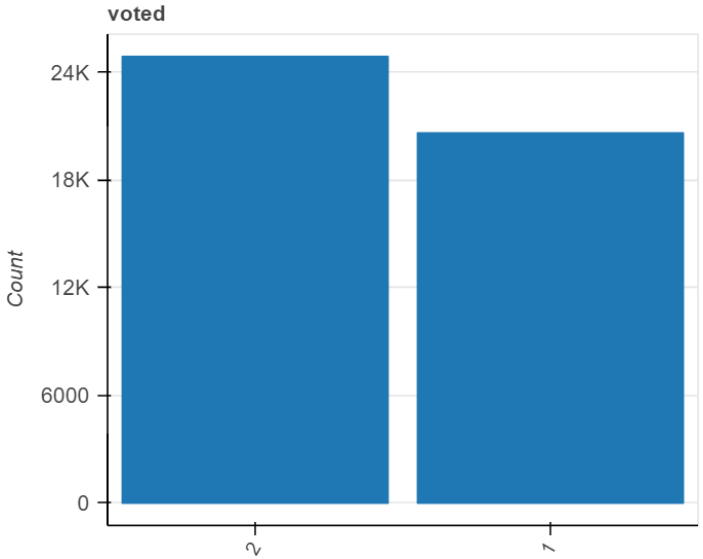
01 Raw Data EDA

Target: Voted

유권자의 지난해 국가 선거 투표 여부 (1 = Yes, 2 = No)

voted
categorical
Show Details

Approximate Distinct Count	2
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Memory Size	2.9 MB




```
[ ]  Counter(train['voted'])
```

```
Counter({2: 24898, 1: 20634})
```

01 Raw Data EDA

응답시간 Outlier

	QaA	QaE	QbA	QbE	QcA	QcE	QdA	QdE	QeA	QeE	QfA	QfE
count	45532.000000	4.553200e+04	45532.000000	4.553200e+04	45532.000000	45532.000000	45532.000000	4.553200e+04	45532.000000	4.553200e+04	45532.000000	4.553200e+04
mean	2.129535	9.453570e+02	2.904463	2.189589e+03	3.662347	1484.294518	1.749078	1.490672e+03	2.317952	1.899292e+03	2.168145	1.850650e+03
std	1.196952	1.307565e+04	1.566142	3.351027e+04	1.431494	8977.664318	1.043625	1.092260e+04	1.369205	1.670765e+04	1.348653	7.608236e+04
min	1.000000	2.500000e+01	1.000000	2.500000e+01	1.000000	25.000000	1.000000	2.600000e+01	1.000000	2.500000e+01	1.000000	2.500000e+01
25%	1.000000	4.040000e+02	1.000000	8.750000e+02	2.000000	651.000000	1.000000	6.790000e+02	1.000000	8.340000e+02	1.000000	5.040000e+02
50%	2.000000	5.570000e+02	3.000000	1.218000e+03	4.000000	899.000000	1.000000	9.310000e+02	2.000000	1.154000e+03	2.000000	7.120000e+02
75%	3.000000	8.270000e+02	4.000000	1.838000e+03	5.000000	1335.000000	2.000000	1.355000e+03	3.000000	1.656000e+03	3.000000	1.078000e+03
max	5.000000	2.413960e+06	5.000000	5.580395e+06	5.000000	871557.000000	5.000000	1.552821e+06	5.000000	1.919926e+06	5.000000	1.176370e+07

응답(Q□A)은 1 to 5 스케일로 이상치 **X,**

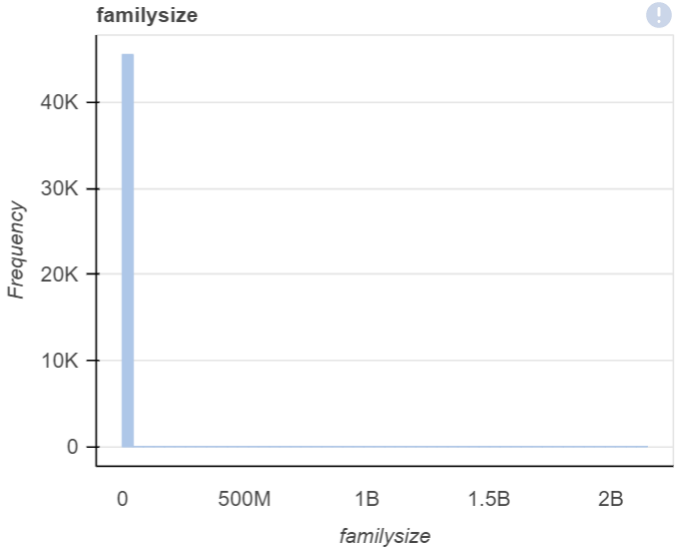
그러나 **응답시간**(Q□E)은 **Q3과 최댓값이 10^3~10^4배 차이,** 이상치 존재 확인 가능

01 Raw Data EDA

형제자매 수 Outlier

familysize
numerical
Show Details

Approximate Distinct Count	27	Mean	47166.8936
Approximate Unique (%)	0.1%	Minimum	0
Missing	0	Maximum	2147483647
Missing (%)	0.0%	Zeros	1217
Infinite	0	Zeros (%)	2.7%
Infinite (%)	0.0%	Negatives	0
Memory Size	711.4 KB	Negatives (%)	0.0%



familysize(형제자매 수) 변수에서도 **현실적으로 불가능한 데이터**(2,147,483,647) 존재,
이상치로 판단

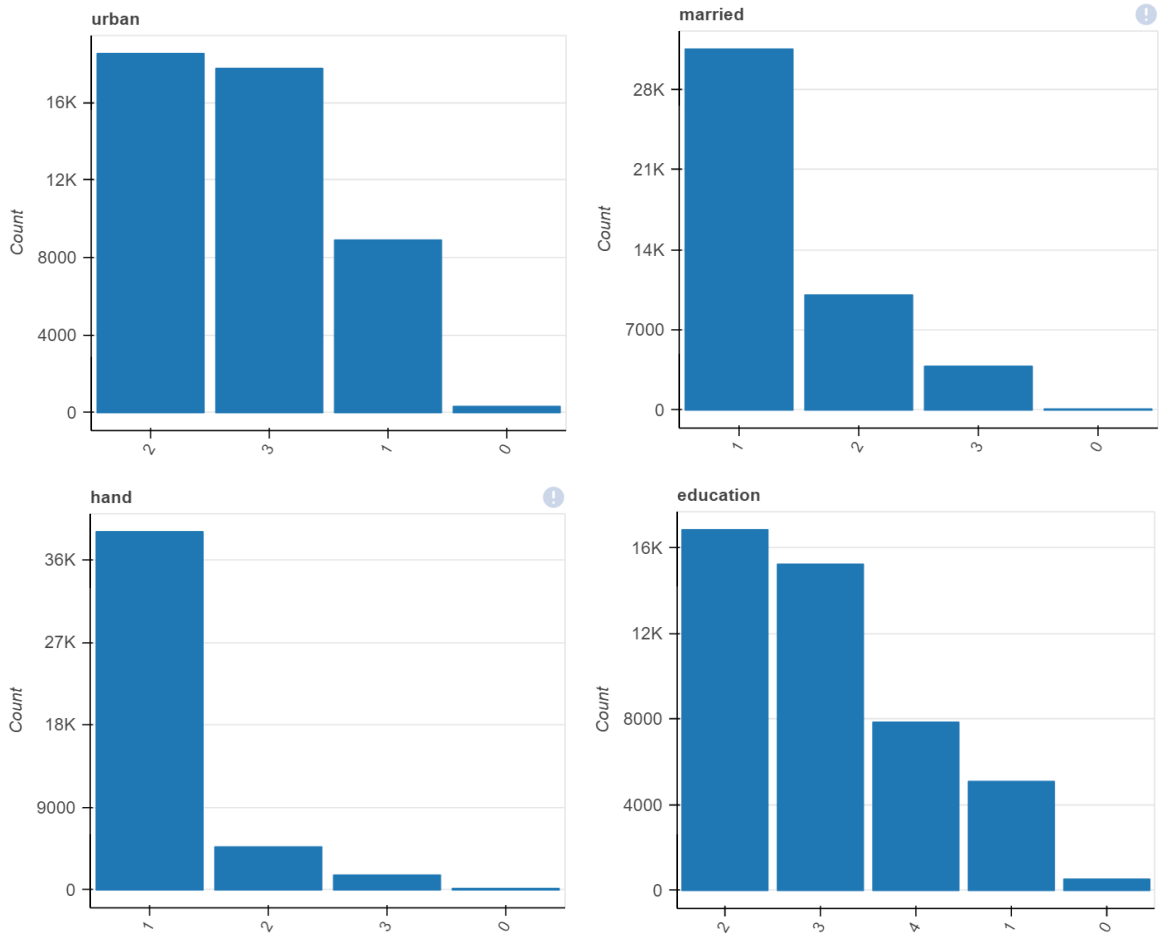
01 Raw Data EDA

무응답 결측치

```
[13] train.isnull().sum()

index    0
QaA      0
QaE      0
QbA      0
QbE      0
        ..
wr_09    0
wr_10    0
wr_11    0
wr_12    0
wr_13    0
Length: 78, dtype: int64
```

데이터 자체에 **결측치(NaN)는 없음,**
그러나 일부 feature에
(education, hand, married, urban)
무응답(=0) 데이터 존재



02

Data Preprocessing

02 Data Preprocessing

Log Transformation 필요성

응답시간(Q□E)과 형제자매 수(familysize)
이상치를 제거하기 위해

3×IQR 범위로 조정해본 결과,

(Min = Q1 − 3×IQR, Max = Q3 + 3×IQR)

Train Set에서 총합 **1만 5천개 이상**의
데이터 손실 발생 (45,532 → 28,699, 16,833 제거됨)

→ 이상치 해소할 다른 방법 필요

```
[21] #아웃라이어 제거할 행 리스트
     out = ['QaE','QbE','QcE','QdE','QeE','QfE','QgE','QhE',
            'QiE','QjE','QkE','QlE','QnE','QmE','QoE','QpE',
            'QqE','QrE','QsE','QtE', 'familysize']
     df[out].describe()
```

```
[22] # 이상치 제거: 통계학에서 사용하는 IQR 기반으로 이상치 제거할 것
     q3 = df[out].quantile(0.75)
     q1 = df[out].quantile(0.25)
     iqr = q3 - q1
     print('iqr', iqr, sep = '₩n')
```

```
[23] def remove_out(dataframe, remove_col):
         dff = dataframe
         for k in remove_col:
             level_1q = q1[k]
             level_3q = q3[k]
             IQR = level_3q - level_1q
             rev_range = 3  # 제거 범위 조절 변수
             dff = dff[(dff[k] <= level_3q + (rev_range * IQR))
             & (dff[k] >= level_1q - (rev_range * IQR))]
             dff = dff.reset_index(drop=True)
         return dff
```

```
[24] df_r = remove_out(df, out)
     df_r[out].describe()
```

```
[19] df_r.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28699 entries, 0 to 28698
Data columns (total 78 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   index       28699 non-null  int64
 1   QaA         28699 non-null  float64
 2   QaE         28699 non-null  int64
```

02 Data Preprocessing

Log Transformation

```
[ ]  # log transform train
     Answers_time_only = ['QaE', 'QbE', 'QcE', 'QdE', 'QeE',
                          'QfE', 'QgE', 'QhE', 'QiE', 'QjE',
                          'QkE', 'QlE', 'QmE', 'QnE', 'QoE',
                          'QpE', 'QqE', 'QrE', 'QsE', 'QtE']

     loged_Answer_time = train[Answers_time_only].copy()
     loged_Answer_time[Answers_time_only] = np.log1p(train[Answers_time_only])
     train[Answers_time_only] = loged_Answer_time
```

```
[ ]  # log transform test
     log_Answer_time_t = test[Answers_time_only].copy()
     log_Answer_time_t[Answers_time_only] = np.log1p(test[Answers_time_only])
     test[Answers_time_only] = log_Answer_time_t
```

응답 시간에(Q□E) 로그 변환 사용

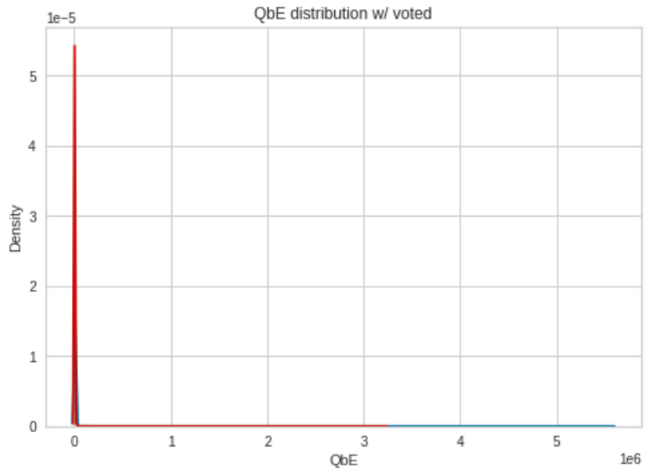
- 비대칭 분포의 넓은 부분은 좁게, 좁은 부분은 넓게 조정
- 10^3~10^4배 수준의 값 차이를 한자릿수 내외로 축소

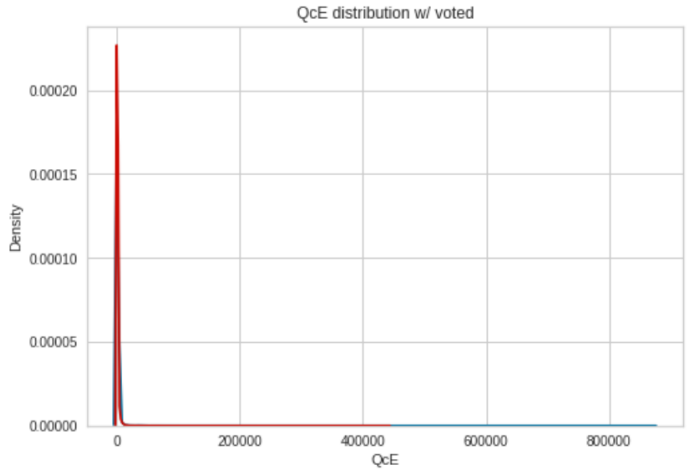
+

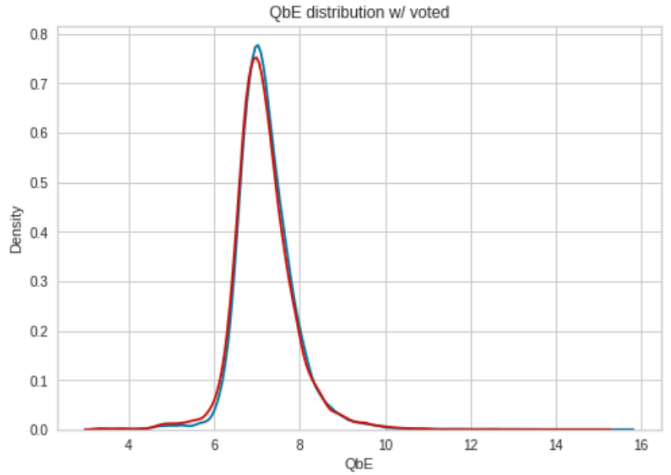
형제자매 수는(familysize)
상식적인 수준에서 범위 조정
(〈20 or 〈100)

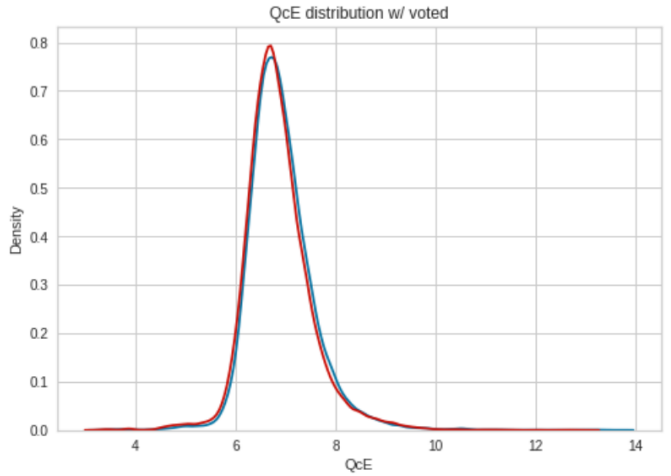
02 Data Preprocessing

Log Transformation: 변환 결과 예시(QbE, QcE)









02 Data Preprocessing

무응답 대치

무응답 있는 Feature의 경우
(education, hand, married, urban)

→ 무응답(0)을 해당
Feature의 중앙값으로 대치

```
[ ]  ## train
     # engnat -> drop ?
     # train = train.drop(train[train.familysize >= 20].index)

     no_ans_col = ['education', 'hand','married' ,'urban' ]

     for i in no_ans_col:
       print(i)
       print(Counter(train[i]))
       train[i] = train[i].replace(0, np.median(train[i]))
       print(Counter(train[i]))
       print()
```

```
education
Counter({2: 16826, 3: 15226, 4: 7853, 1: 5088, 0: 527})
Counter({2: 16826, 3: 15753, 4: 7853, 1: 5088})

hand
Counter({1: 39049, 2: 4691, 3: 1619, 0: 161})
Counter({1: 39210, 2: 4691, 3: 1619})

married
Counter({1: 31543, 2: 10057, 3: 3827, 0: 93})
Counter({1: 31636, 2: 10057, 3: 3827})

urban
Counter({2: 18529, 3: 17764, 1: 8905, 0: 322})
Counter({2: 18851, 3: 17764, 1: 8905})
```

```
[ ]  ## test
     no_ans_col = ['education', 'hand','married' ,'urban' ]

     for i in no_ans_col:
       print(i)
       print(Counter(test[i]))
       test[i] = test[i].replace(0, np.median(test[i]))
       print(Counter(test[i]))
       print()
```

```
education
Counter({2: 4236, 3: 3798, 4: 1963, 1: 1264, 0: 122})
Counter({2: 4236, 3: 3920, 4: 1963, 1: 1264})

hand
Counter({1: 9818, 2: 1155, 3: 370, 0: 40})
Counter({1: 9858, 2: 1155, 3: 370})

married
Counter({1: 7928, 2: 2469, 3: 958, 0: 28})
Counter({1: 7956, 2: 2469, 3: 958})

urban
Counter({2: 4551, 3: 4439, 1: 2299, 0: 94})
Counter({2: 4645, 3: 4439, 1: 2299})
```

02 Data Preprocessing

PCA

주성분 분석 결과,

46개 주성분으로

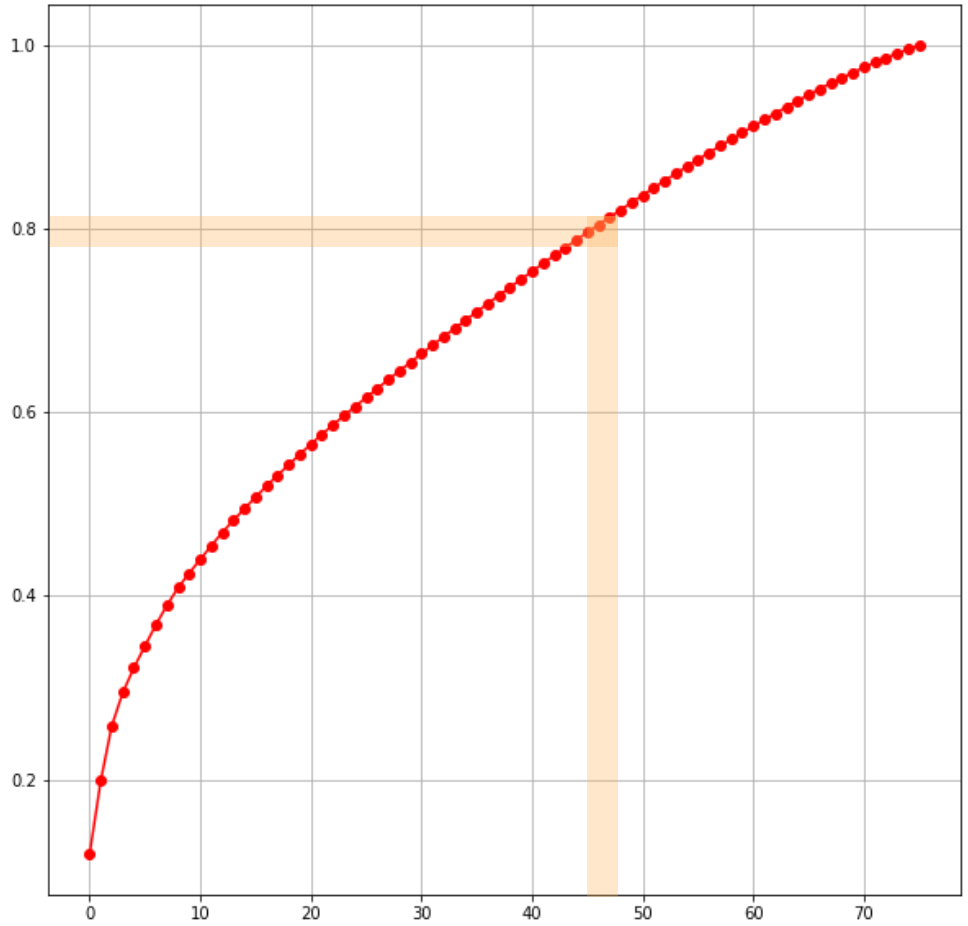
전체 분산의 80% 설명 가능

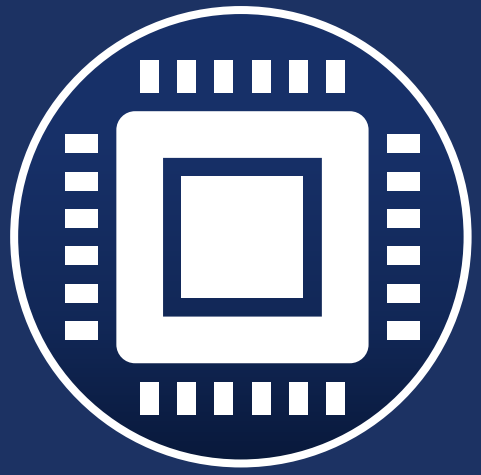
```
[ ]  from sklearn.decomposition import PCA

     pca = PCA(random_state=0)
     x_pca = pca.fit_transform(x_train_scaled)
     pd.Series(np.cumsum(pca.explained_variance_ratio_))

     # 전체 분산의 80% 이상을 설명하는 46개 feature로 축소할 수 있음
```

```
0     0.118855
1     0.198474
2     0.258546
      ...
44    0.787750
45    0.796097
46    0.804333
47    0.812506
48    0.820627
```





03
Modeling

03 Modeling

모델링 케이스 3가지

1. 전처리 X, AutoML

- 전처리 없이 AutoML 적용
- LightGBM, CatBoost, GBC
- 상위 3개 모델 Ensemble

2. 전처리 & 직접 모델링

- Log Transformation, 결측치(무응답) 대치, PCA, label-encoding 처리
- RF, GB, LGBM, LDA 모델
- Grid Search&Ensemble

3. 전처리 & AutoML

- Log Transformation, 무응답 대치 처리
- CatBoost, LightGBM, GBC
- 상위 3개 모델 Ensemble

03 Modeling Case 1.

전처리 X, AutoML

전처리의 성능 영향 확인하기 위해

데이터 전처리 없이 바로 AutoML 사용

상위 3개 모델 Ensemble해 성능 도출
(LightGBM, CatBoost, GBC)

Train 내부 성능 0.7661

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.6950	0.7666	0.6417	0.7618	0.6966	0.3945	0.4005	0.836
catboost	CatBoost Classifier	0.6958	0.7658	0.6562	0.7544	0.7019	0.3944	0.3985	17.529
gbc	Gradient Boosting Classifier	0.6960	0.7645	0.6393	0.7650	0.6965	0.3968	0.4033	16.728
lda	Linear Discriminant Analysis	0.6929	0.7621	0.6628	0.7460	0.7019	0.3874	0.3904	0.760
et	Extra Trees Classifier	0.6941	0.7620	0.6432	0.7594	0.6964	0.3923	0.3979	6.943
rf	Random Forest Classifier	0.6873	0.7561	0.6323	0.7549	0.6881	0.3794	0.3855	6.660
ada	Ada Boost Classifier	0.6897	0.7556	0.6513	0.7477	0.6961	0.3822	0.3861	3.151
xgboost	Extreme Gradient Boosting	0.6792	0.7484	0.6646	0.7247	0.6933	0.3583	0.3598	14.217

```
# blended = stack_models(estimator_list = best_3)
blended = blend_models(estimator_list = best_3, fold = 5, method = 'soft')
```

```
pred_holdout = predict_model(blended)
# blended
```

```
INFO:logs:Initializing predict_model()
INFO:logs:predict_model(estimator=VotingClassifier(estimators=[('lightgbm',
...
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Voting Classifier	0.6944	0.7661	0.6394	0.7659	0.6969	0.3938	0.4004

03 Modeling Case 2.

전처리 & 직접 모델링

Train set 내부에서 다시 샘플링한 뒤,

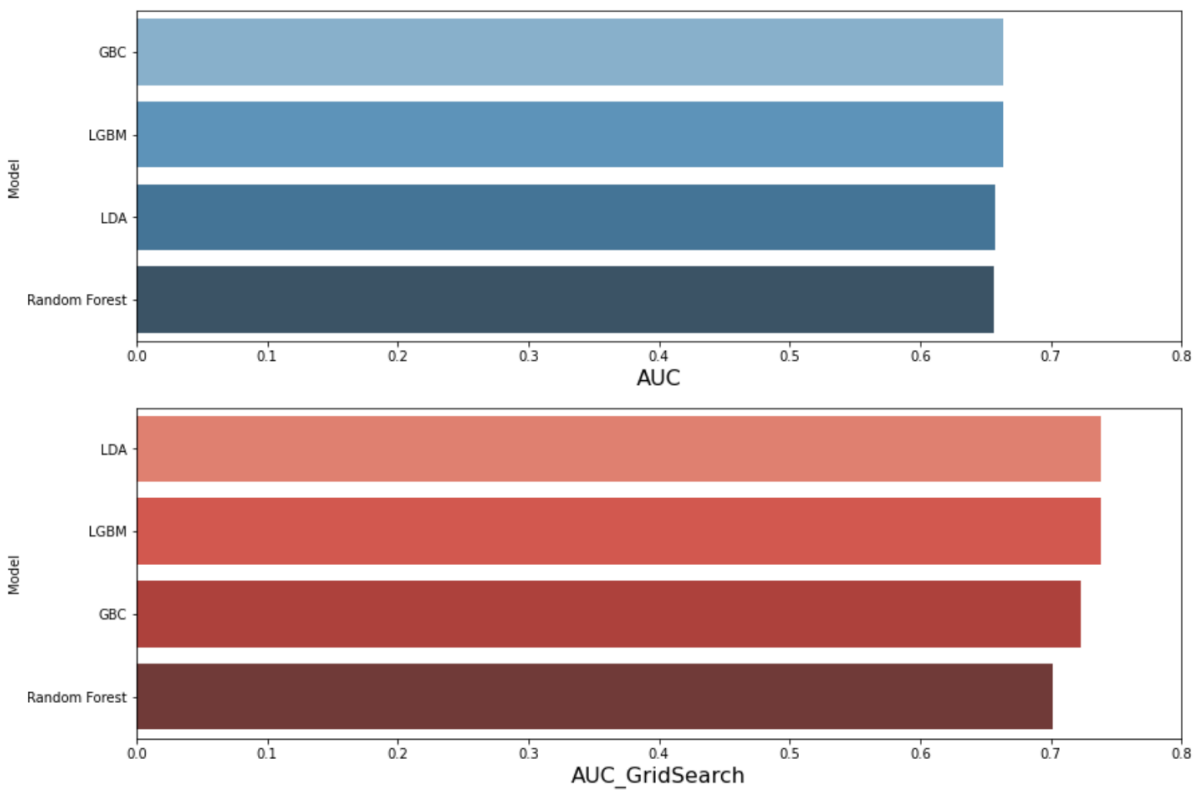
데이터 전처리 (Log Transformation,

결측치(무응답) 대치, PCA, label-encoding)

Sklearn 라이브러리 이용해 모델링,
(랜덤 포레스트, GBC, LGBM, LDA)

GridSearch 성능 개선 및 Ensemble

Train 내부 성능 0.7364




```
# soft-voting
GBC_E = GradientBoostingClassifier(learning_rate = 0.1, loss = 'deviance', max_depth = 4, max_features = 0.3, min_samples_leaf = 50, n_estimators = 30)
LGBM_E = LGBMClassifier(learning_rate = 0.1, max_depth = -1, min_child_samples = 5, num_leaves = 10, reg_alpha = 0.03)
LDA_E = LinearDiscriminantAnalysis(solver = 'svd')

VC = VotingClassifier(estimators=[('gbc_e',GBC_E),('lgbm_e',LGBM_E),('lda_e', LDA_E)], voting = 'soft')
VC_final = VC.fit(x_train2, y_train2)
```

```
pred_proba = VC_final.predict_proba(x_test2)
AUC = roc_auc_score(y_test2, pred_proba[:, 1])
AUC
```

```
0.736387236545986
```

03 Modeling Case 3.

전처리 & AutoML

데이터 전처리
(Log Transformation, 결측치(무응답) 대치)

단, PCA, label-encoding 사용 X

상위 3개 모델 Ensemble해 성능 도출
(CatBoost, LightGBM, GBC)

Train 내부 성능 0.7698

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
catboost	CatBoost Classifier	0.6933	0.7647	0.6587	0.7514	0.7019	0.3889	0.3926	24.771
lightgbm	Light Gradient Boosting Machine	0.6934	0.7636	0.6462	0.7589	0.6979	0.3907	0.3959	1.442
gbc	Gradient Boosting Classifier	0.6943	0.7634	0.6454	0.7608	0.6983	0.3926	0.3981	22.305
lda	Linear Discriminant Analysis	0.6907	0.7601	0.6582	0.7475	0.7000	0.3834	0.3868	1.503
lr	Logistic Regression	0.6908	0.7594	0.6619	0.7457	0.7012	0.3831	0.3862	9.112
et	Extra Trees Classifier	0.6895	0.7581	0.6450	0.7533	0.6949	0.3827	0.3875	9.321
ada	Ada Boost Classifier	0.6898	0.7563	0.6519	0.7498	0.6973	0.3824	0.3864	4.425
rf	Random Forest Classifier	0.6850	0.7516	0.6354	0.7518	0.6887	0.3744	0.3799	9.248

```
blended = blend_models(estimator_list = best_3, fold = 5, method = 'soft')
```

```
pred_holdout = predict_model(blended)
```

```
INFO:logs:Initializing predict_model()
INFO:logs:predict_model(estimator=VotingClassifier(estimators=[('catboost',
```

...

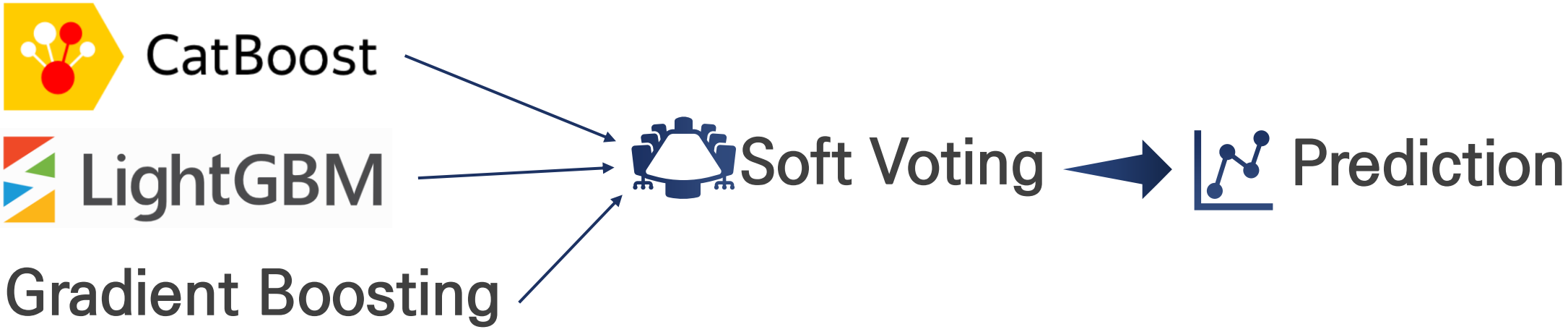
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Voting Classifier	0.6974	0.7698	0.6438	0.7695	0.7011	0.3997	0.4062

04
Result

04 Result

Train 결과 해석

Train 내부 성능이 가장 좋은 **Case 3. 전처리 & AutoML**의 모델:



- **Boosting 기반 모델**의 전반적으로 높은 성능
- **범주형 Feature에 강한 CatBoost**가 가장 높은 적합도
- 모델마다 가중치를 두는 **Soft Voting이 Ensemble로 적합**

04 Result

Test set 예측 제출

Train 내부 성능이 가장 좋은

Case 3. 전처리 & AutoML의 모델로

Test set 예측, 최종 **제출 및 평가**

```
final_model = finalize_model(blended)
```

```
predictions = predict_model(final_model, data = test)

submission['voted'] = predictions['Label']

submission.to_csv(path+'/submission_proba(log_out_change).csv')
```

```
INFO:logs:Initializing predict_model()
INFO:logs:predict_model(estimator=VotingClassifier(estimators=[('catboost',
```

722478	submission_proba(log_out_change).csv edit	2022-08-30 15:22:45	0.7055298984 0.7129010842

최종 Test set 성능(AUC): **0.7055 ~ 0.7190%**

04 Result

의의와 한계

무응답 등 isnull()에 잡히지 않는 **실질적 결측치 포착 및 조치**

그러나 EDA 과정에서 도메인 지식 반영에 한계 존재

로그 변환을 통해 관측치를 최대한 보존하면서

특정 Feature들의 이상치, Skewness 문제 개선

전처리 여부, AutoML 혹은 Sklearn을 이용한 직접 모델링 등

다양한 조건에서 접근 시도

이상으로 발표를 마칩니다.

감사합니다!