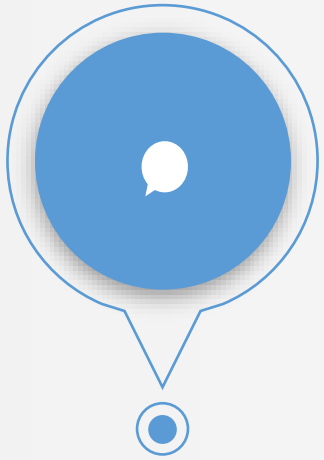
22-2 머신러닝 분반

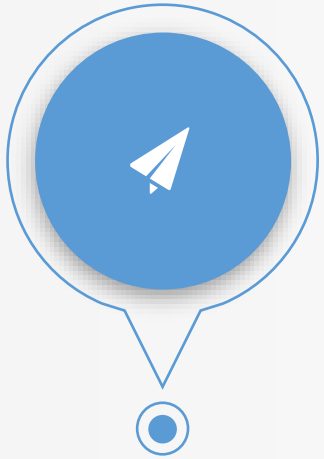
심리 성향 예측 AI 경진대회

16기 노연수
16기 유우혁
16기 박종혁
16기 천원준

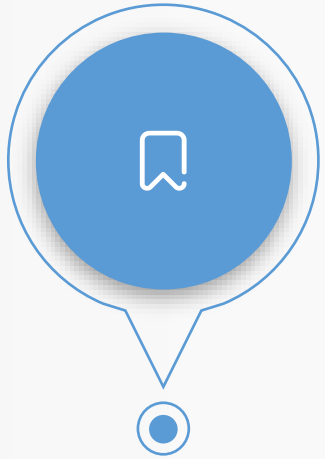
Contents



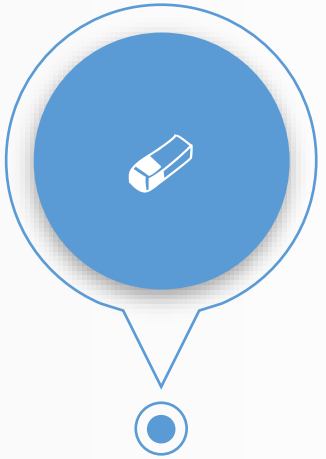
데이터 소개



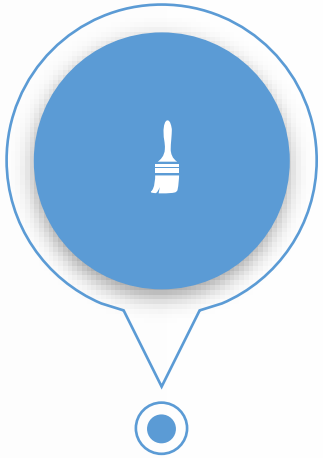
EDA



Data Preprocessing



Modeling



Conclusion

데이터 소개

심리 성향 테스트를 활용해 설문자의 국가 선거 투표 여부를 맞추는 알고리즘 개발

심사 기준 : AUC

Target :

voted(지난 해 국가 선거 투표 여부)

1 = Yes, 2 =No

변수 설명 EDA

Mean encoding이란?

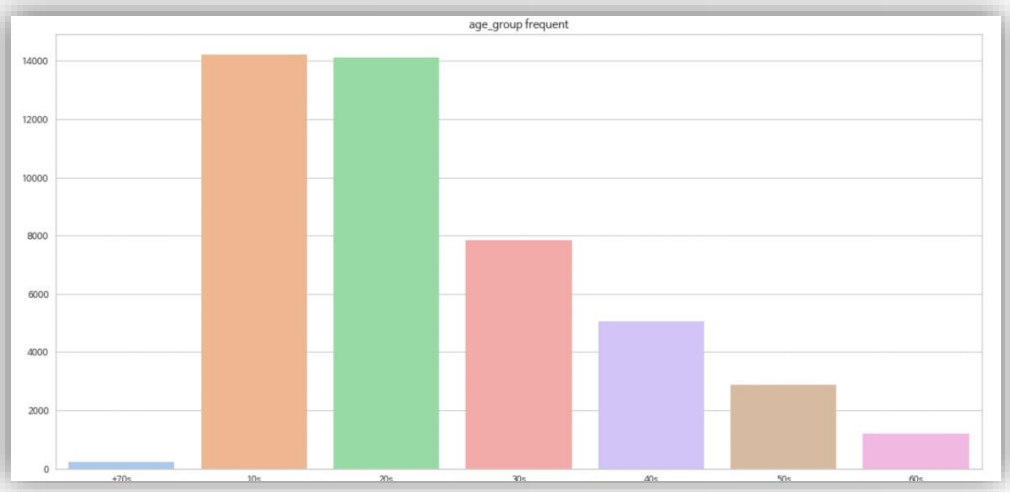
	familysize
0	0.413311
1	0.424331
2	0.450106
3	0.456912
4	0.483391
5	0.481651
6	0.464200
7	0.519380
8	0.438914
9	0.476190
10	0.521212

Family size mean encoding

- Mean Encoding 은 구분을 넘어 좀 더 의미 있는 Encoding 을 하려는 시도
- Encoding 하는 Feature와 예측 하려고 하는 Target 간의 어떤 수치적인 관계를 Categorical 에서 찾으려는 노력

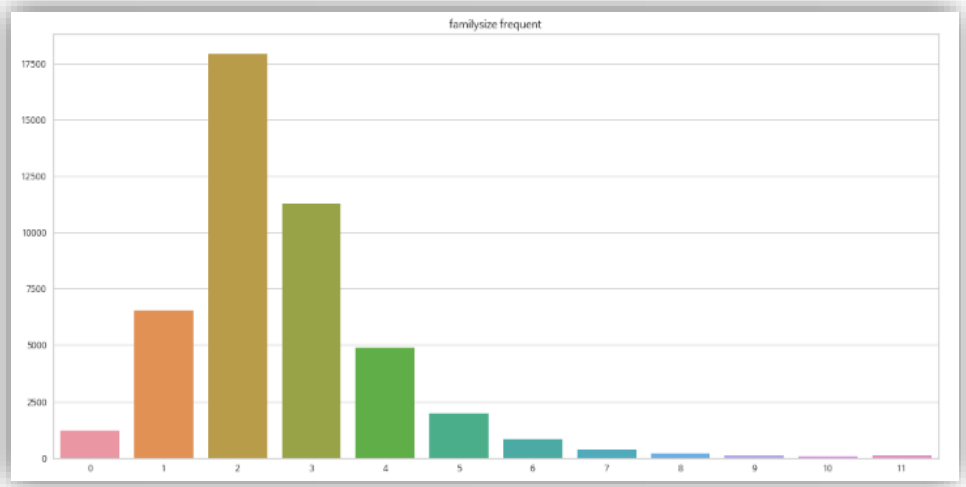
변수 설명 EDA

Age group



60대와 70대 데이터 병합

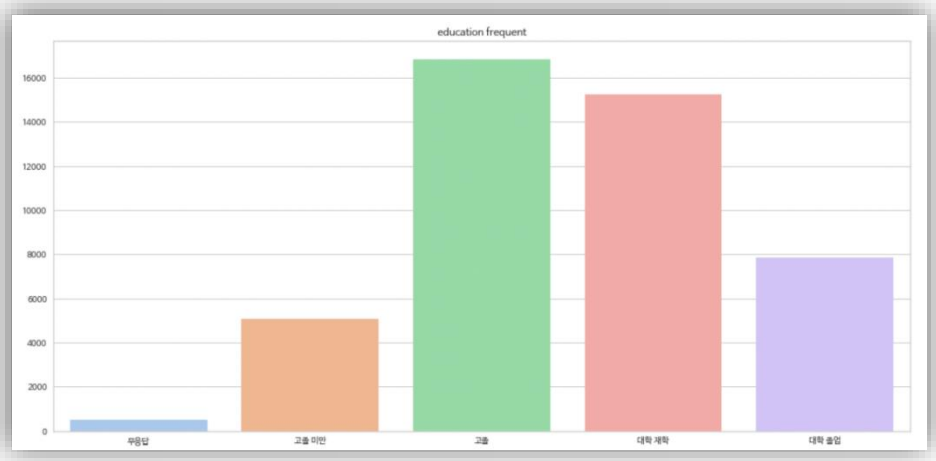
Family size



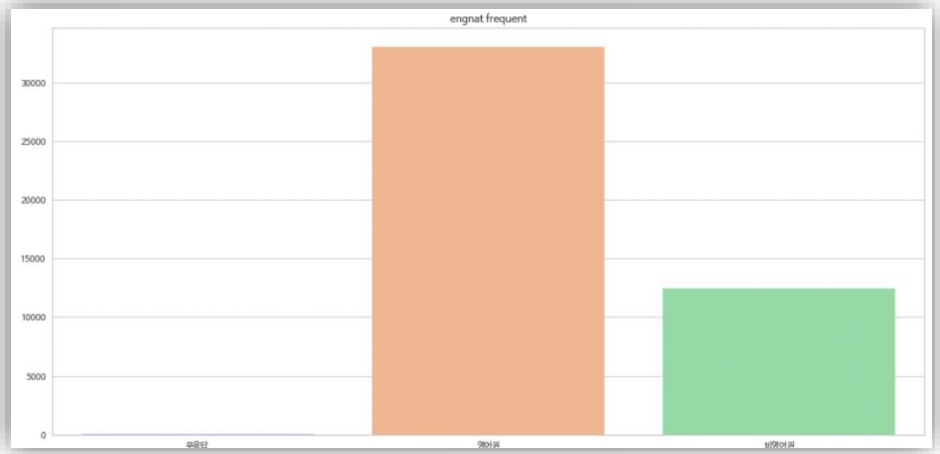
11명 이상인 데이터 병합

변수 설명 EDA

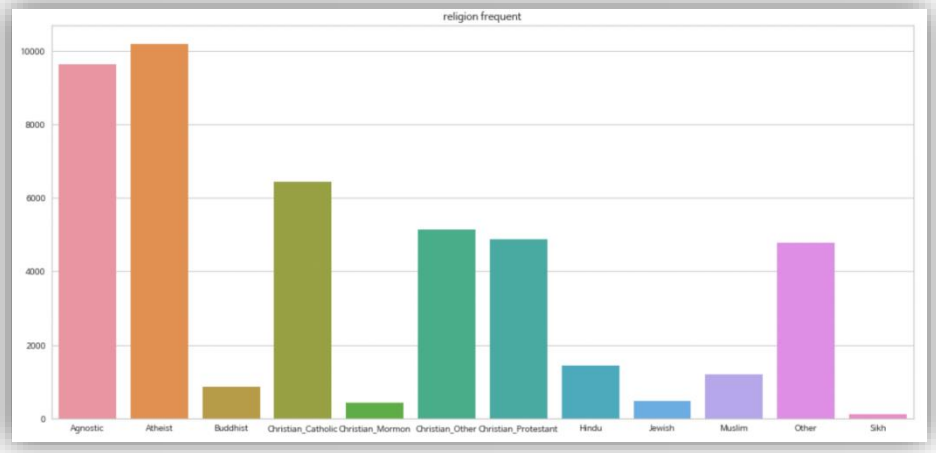
Education



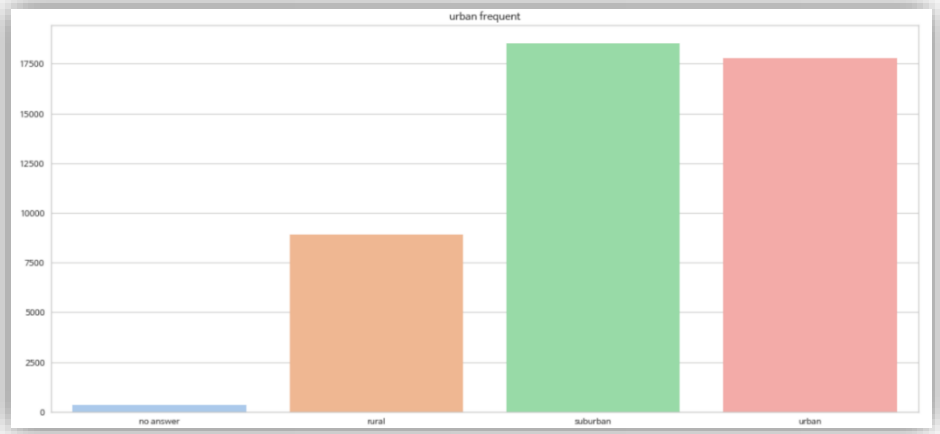
Engnat



Religion

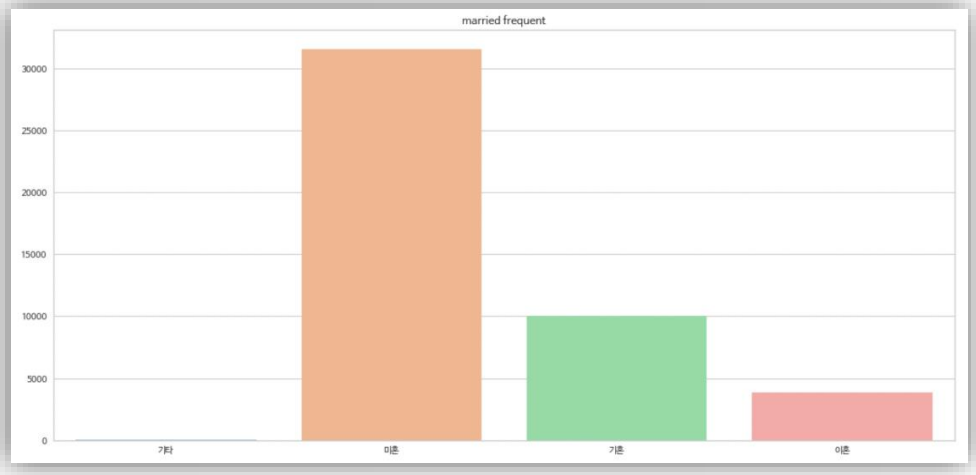


Urban

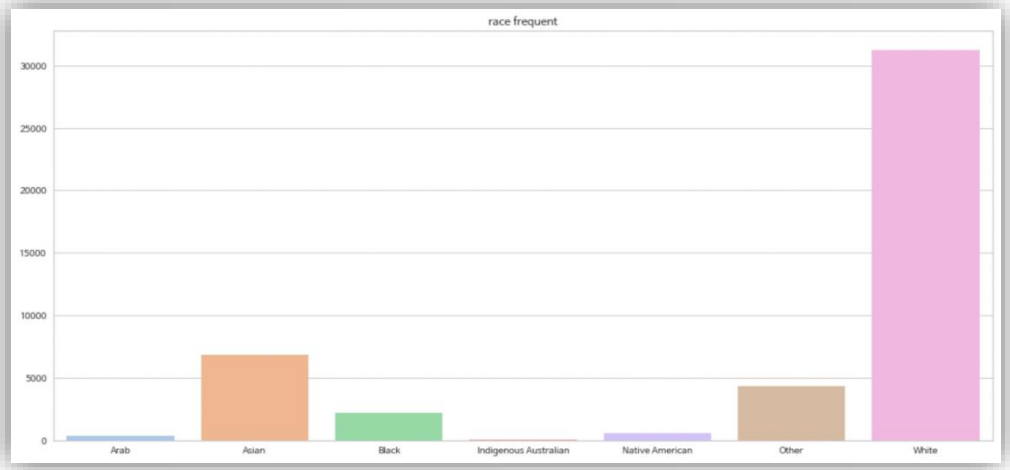


변수 설명 EDA

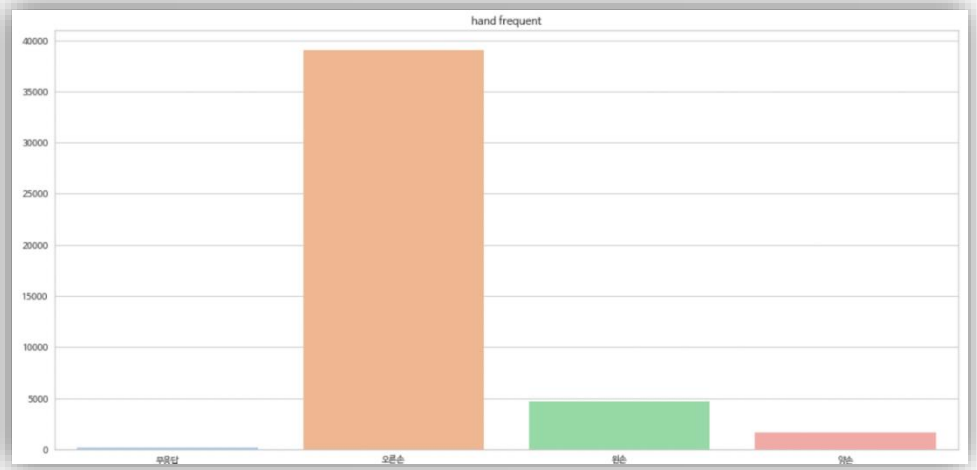
Married



Race

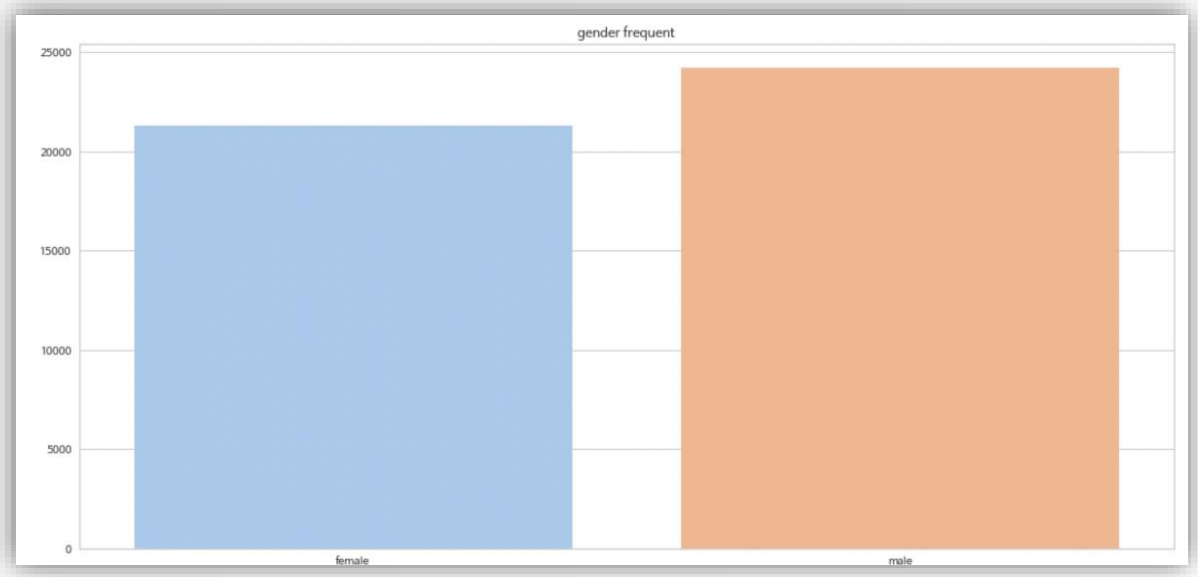


Hand



변수 설명 EDA

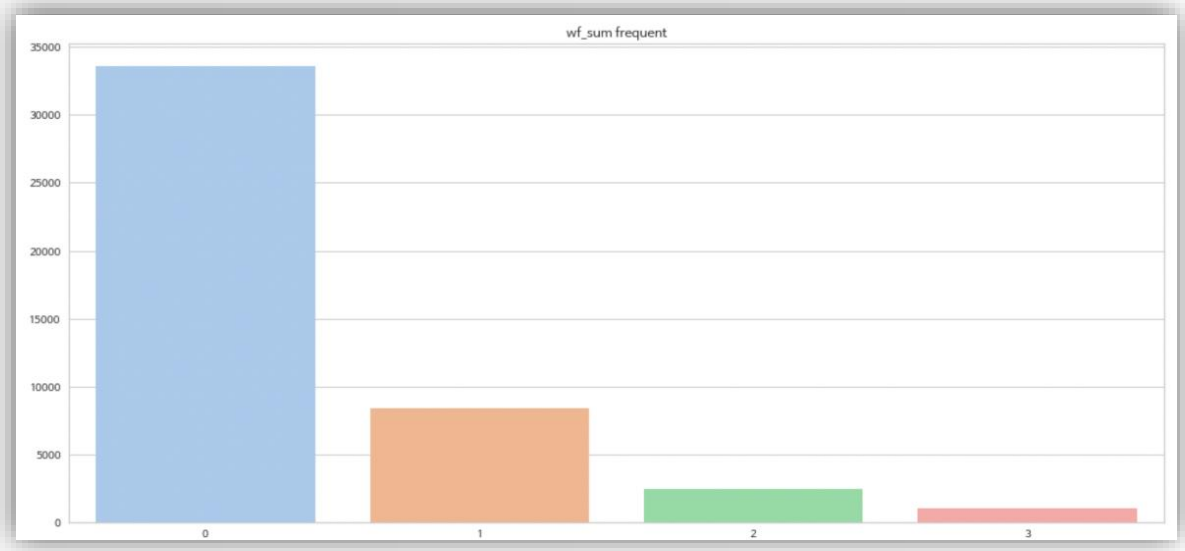
Gender




```
encoded = encoder.fit_transform(train[['gender']]).toarray()

final_train['gender_0'] = encoded[: , 0]
final_train['gender_1'] = encoded[: , 1]

final_train = final_train.drop('gender', axis = 1)
```

변수 설명 EDA

Wf_




```
for column in wf_column:
  encoder = OneHotEncoder()
  encoded = encoder.fit_transform(train[[column]]).toarray()
  final_train[column + '_0'] = encoded[: , 0]
  final_train[column + '_1'] = encoded[: , 1]
  final_train = final_train.drop(column, axis = 1)
  encoded = encoder.transform(test[[column]]).toarray()
  final_test[column + '_0'] = encoded[: , 0]
  final_test[column + '_1'] = encoded[: , 1]
  final_test = final_test.drop(column, axis = 1)
```

파생 변수 생성

Q_A (마키아벨리즘 테스트)

데이터상 코드	문항	토론 게시물상 질문 번호	Subscale	부호
Qa	?	?	?	?
Qb	The biggest difference between most criminals and other people is that the criminals are stupid enough to get caught.	Q11	Views	+
Qc	Anyone who completely trusts anyone else is asking for trouble.	Q1	Tactics	+
Qd	?	?	?	?
Qe	P.T. Barnum was wrong when he said that there's a sucker born every minute.	Q13	Views	-
Qf	There is no excuse for lying to someone else.	Q19	Tactics	-
Qg	?	?	?	?
Qh	Most people forget more easily the death of their parents than the loss of their property.	Q17	Views	+
Qi	?	?	?	?
Qj	It is safest to assume that all people have a vicious streak and it will come out when they are given a chance.	Q3	Views	+
Qk	All in all, it is better to be humble and honest than to be important and dishonest.	Q20	Morality	-
Ql	?	?	?	?
Qm	It is hard to get ahead without cutting corners here and there.	Q7	Views	+
Qn	?	?	?	?
Qo	The best way to handle people is to tell them what they want to hear.	Q10	Tactics	+
Qp	?	?	?	?
Qq	Most people are basically good and kind.	Q14	Views	-
Qr	One should take action only when sure it is morally right.	Q4	Tactics	-
Qs	It is wise to flatter important people.	Q6	Tactics	+
Qt	?	?	?	?

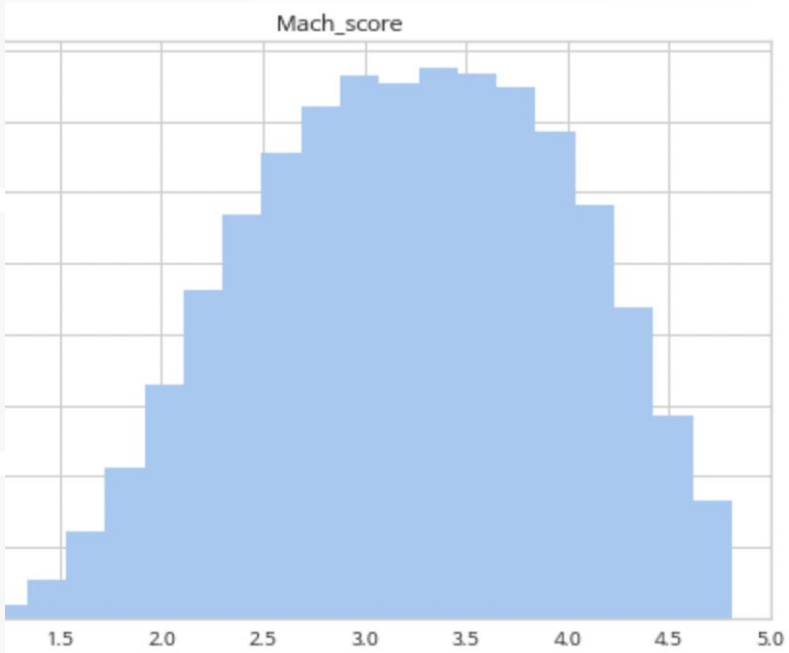
파생 변수 생성

Q_A (마키아벨리즘 테스트)

```
[ ]  # secret이 아닌 질문들 중 상관계수가 양의 질문들과 상관계수가 -인 column들

     reverse_know_columns = ['QeA', 'QfA', 'QkA', 'QqA', 'QrA']
     for c in reverse_know_columns:
       train_QA[c] = 6- train_QA[c]
```

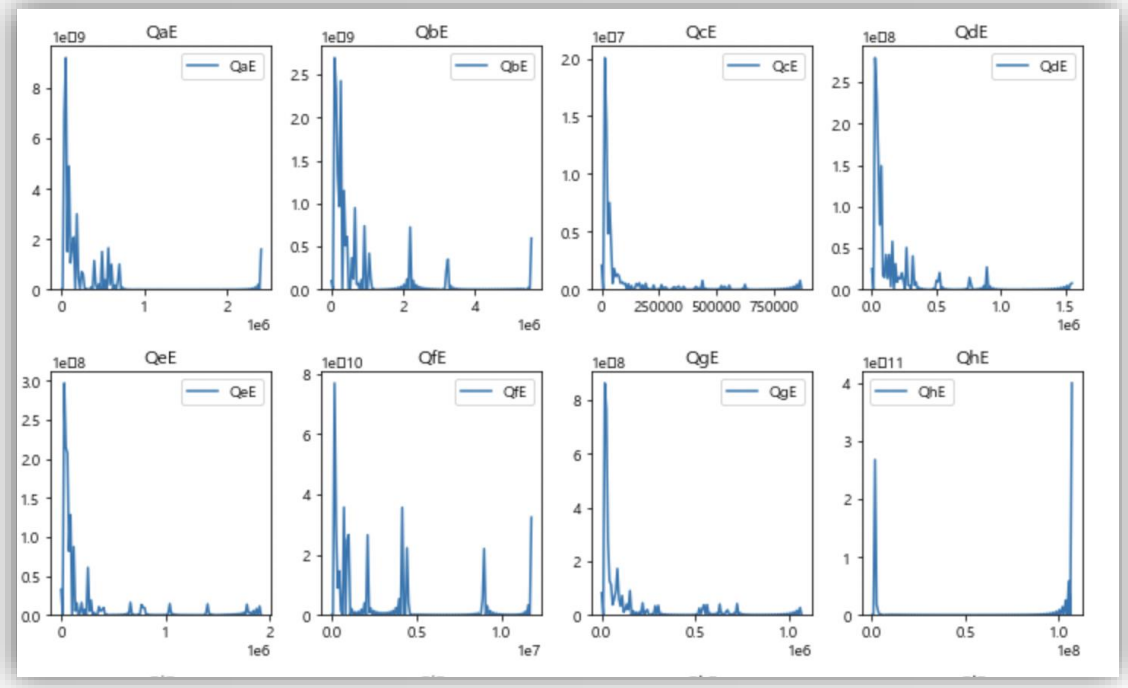
```
[ ]  # secret 질문들 중 상관계수가 부호가 양의 질문들과 음으로 나와 -로 예상되는 column들

     reverse_unknow_columns = ['QaA', 'QdA', 'QgA', 'QiA', 'QnA']
     for c in reverse_unknow_columns:
       train_QA[c] = 6- train_QA[c]
```

```
[ ]  # 마키아밸리즘 스코어를 평균 내린 mach_score 변수 생성

     train_QA['Mach_score'] = train_QA.mean(axis = 1)

     final_train['Mach_score'] = train_QA['Mach_score']
     final_train.drop(QA_columns, axis = 1, inplace = True)
```



데이터 전처리

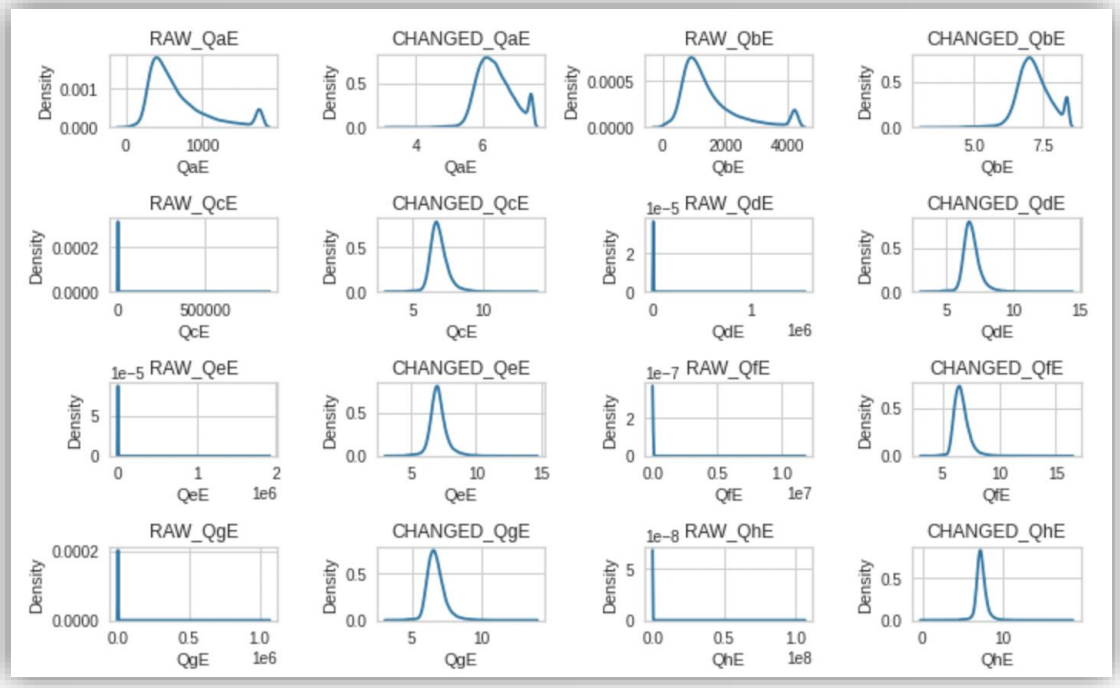
Q_E




```
# 95%이상의 데이터는 95%값으로 교체하기
# EDA에서 살펴본 column들 변환

outlier_columns = ['QaE','QbE','QqE','QrE']

changed_train_QE = train_QE.loc[:,outlier_columns].apply(lambda x: x.clip(x.quantile(.00), x.quantile(.95)), axis=0)
train_QE.loc[:,outlier_columns] = changed_train_QE
```

데이터 전처리

Q_E




```
[ ]  # 데이터의 스케일이 너무 커서 로그변환 적용하기

     changed_train_QE = np.log1p(train_QE)
     train_QE.loc[: , QE_columns] = changed_train_QE
     final_train.loc[: , QE_columns] = train_QE
```

파생 변수 생성

Tp__

1. ___ 나는 활발하고 열심히 하는 사람이다.
2. ___ 나는 따지기를 좋아하고 다투기를 잘하는 사람이다.
3. ___ 나는 믿음직스럽고 자기관리가 가능한 사람이다.
4. ___ 나는 불안하고 화를 잘 내는 사람이다.
5. ___ 나는 새로운 경험을 마다하지 않으며 여러 가지로 생각해보는 사람이다.
6. ___ 나는 내향적이고 조용한 사람이다.
7. ___ 나는 동정심이 많고 다정한 사람이다.
8. ___ 나는 계획적이지 않고 조심성 없는 사람이다.
9. ___ 나는 침착하고 기분이 안정된 사람이다.
10. ___ 나는 변화를 싫어하며 창의적이지 않은 사람이다.

1 전혀 그렇지 않다

2 어느 정도 그렇지 않다

3 약간 그렇지 않다

4 그럴 수도, 아닐 수도 있다

5 약간 그렇다

6 어느 정도 그렇다

7 매우 그렇다

파생 변수 생성

Tp__

성실성 : {'3번 점수'+(8-'8번 점수')}÷2

친화성 : {'7번 점수'+(8-'2번 점수')}÷2

정서적 안정성 : {'9번 점수'+(8-'4번 점수')}÷2

-점수가 낮다면 신경성과 관련이 있습니다.

경험 개방성 : {'5번 점수'+(8-'10번 점수')}÷2

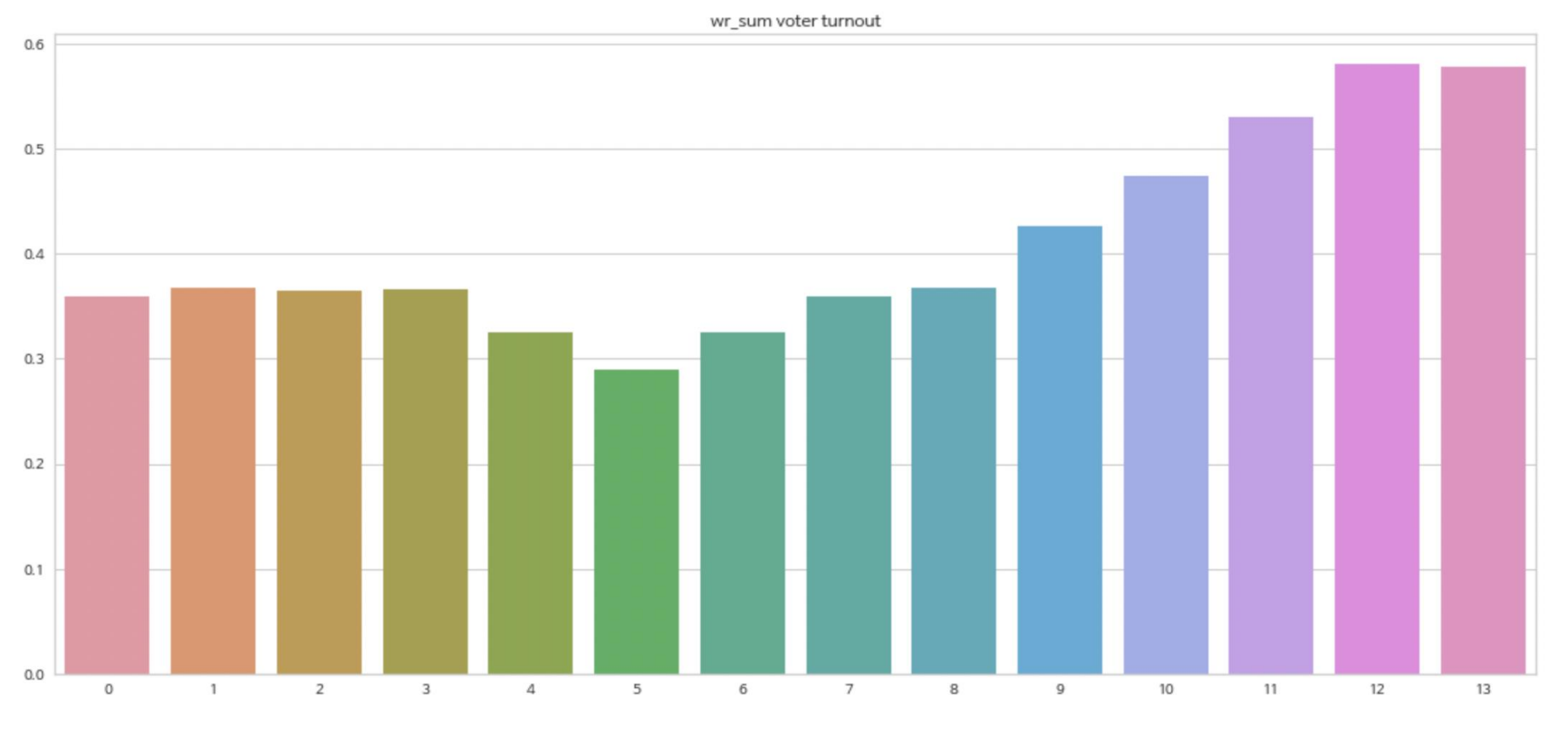
외향성 {'1번 점수'+(8-'6번 점수')}÷2

```
train['Extraversion']=(train['tp01']+train['tp06'])/2
train['Agreeableness']=(train['tp02']+train['tp07'])/2
train['Conscientiousness']=(train['tp03']+train['tp08'])/2
train['Emotional Stability']=(train['tp04']+train['tp09'])/2
train['Openness to Experiences']=(train['tp05']+train['tp10'])/2
```

파생 변수 생성

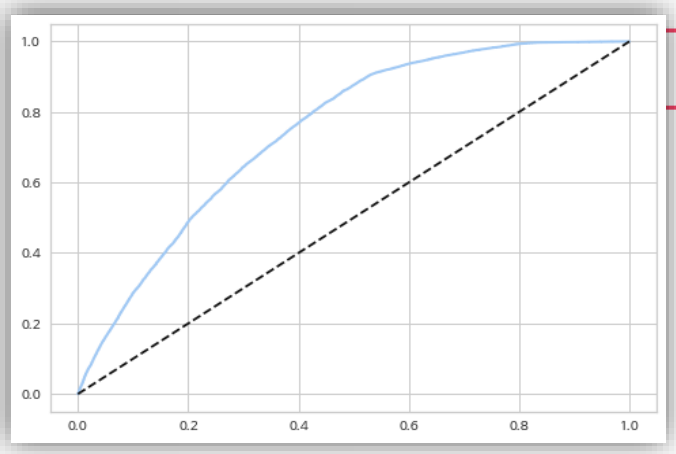
Wr_

단어의 정의를 알고 있는지 확인: 1=Yes, 0=No
총 13개의 단어 중 몇개의 단어를 알고 있는지 Sum column으로 변환



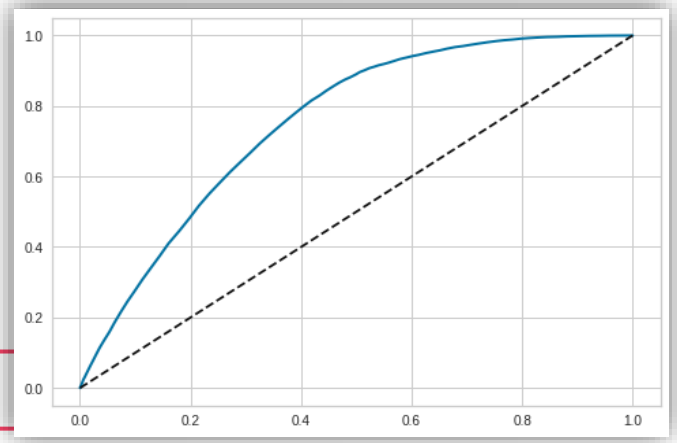
Modeling

<LogisticRegression>



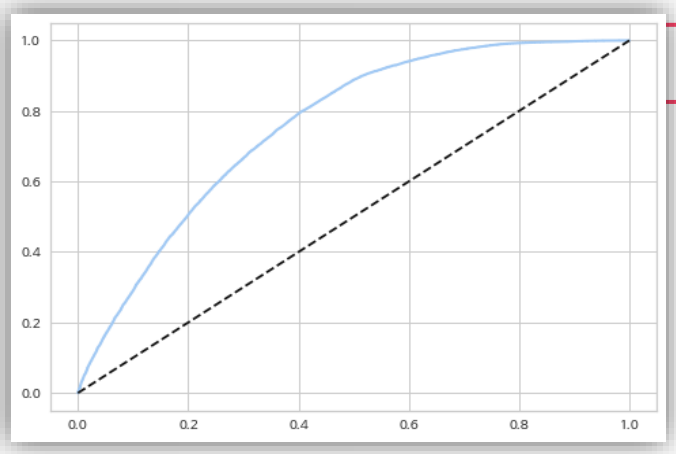
AUC : 0.7508

<RandomForestClassifier>



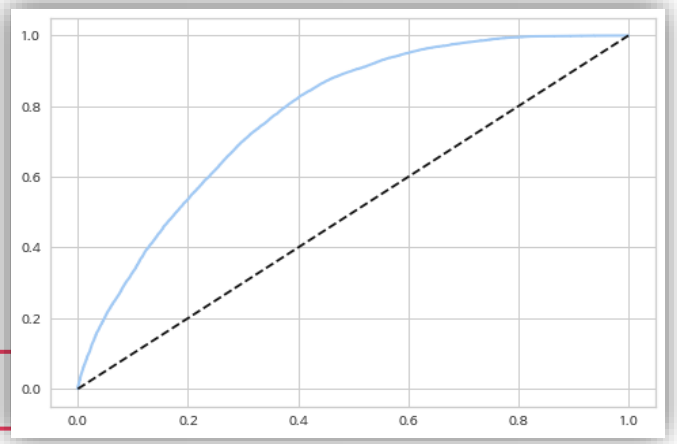
AUC : 0.7533

<LinearDiscriminantAnalysis>



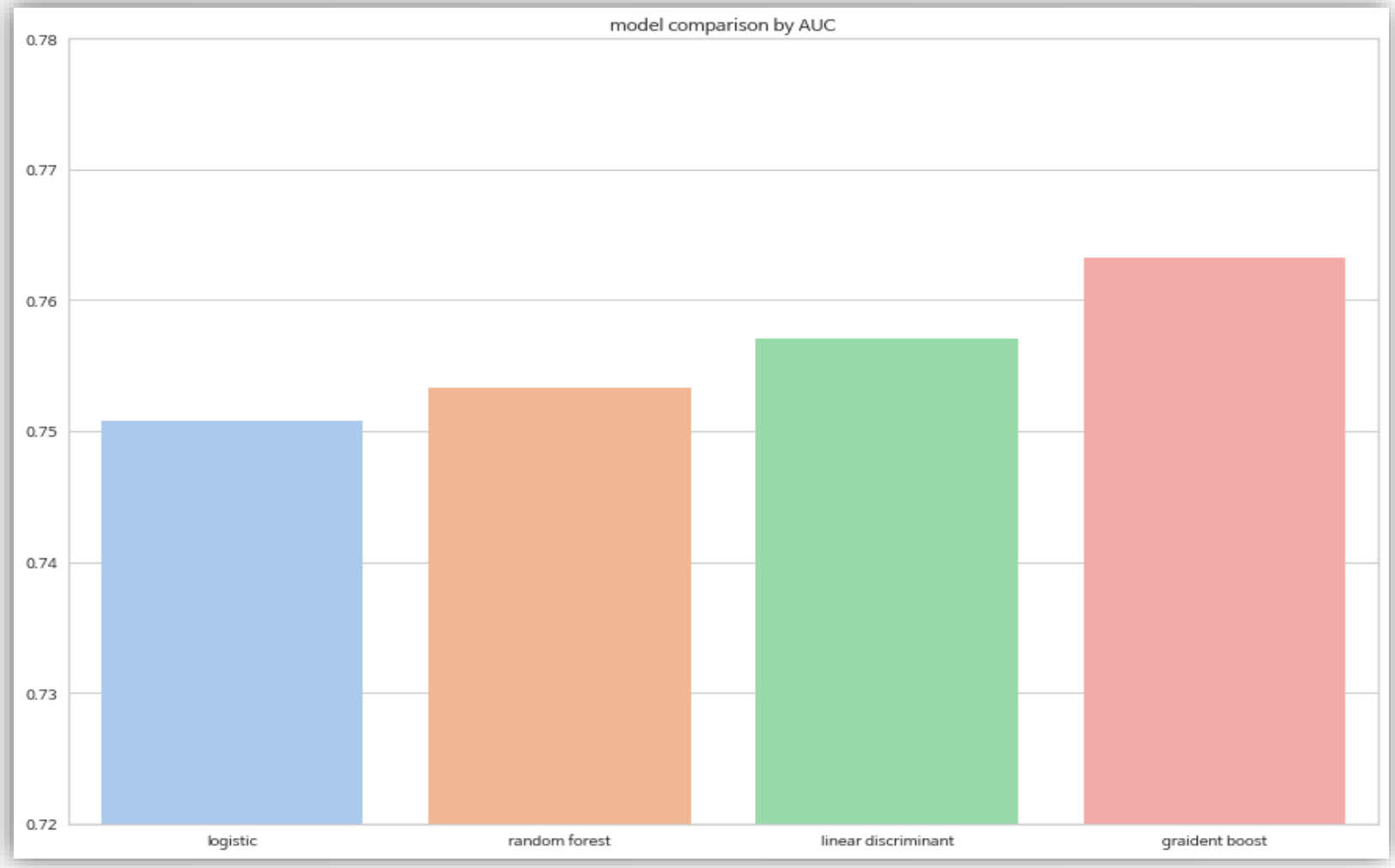
AUC : 0.7571

<GradientBoostingClassifier>



AUC : 0.7632

Modeling
- 모델 비교



Modeling

– Auto ML

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.691	0.761	0.746	0.636	0.686	0.386	0.391	14.200
lightgbm	Light Gradient Boosting Machine	0.690	0.759	0.751	0.633	0.687	0.384	0.390	1.004
catboost	CatBoost Classifier	0.688	0.758	0.735	0.635	0.681	0.379	0.383	20.520
lda	Linear Discriminant Analysis	0.684	0.757	0.715	0.635	0.672	0.370	0.373	0.308
lr	Logistic Regression	0.684	0.756	0.690	0.640	0.664	0.366	0.367	5.670
et	Extra Trees Classifier	0.686	0.754	0.723	0.635	0.676	0.374	0.377	5.178
ada	Ada Boost Classifier	0.687	0.752	0.733	0.633	0.679	0.376	0.380	2.886
rf	Random Forest Classifier	0.682	0.751	0.720	0.631	0.672	0.367	0.370	9.320
xgboost	Extreme Gradient Boosting	0.674	0.741	0.689	0.627	0.656	0.347	0.348	8.514
nb	Naive Bayes	0.666	0.716	0.765	0.604	0.675	0.342	0.352	0.086
dt	Decision Tree Classifier	0.610	0.607	0.568	0.570	0.569	0.213	0.213	0.952
qda	Quadratic Discriminant Analysis	0.544	0.580	0.439	0.542	0.400	0.071	0.100	0.168
knn	K Neighbors Classifier	0.556	0.570	0.544	0.509	0.526	0.109	0.109	48.116
dummy	Dummy Classifier	0.547	0.500	0.000	0.000	0.000	0.000	0.000	0.062
svm	SVM - Linear Kernel	0.614	0.000	0.448	0.514	0.430	0.199	0.235	1.158
ridge	Ridge Classifier	0.684	0.000	0.714	0.635	0.672	0.369	0.372	0.078

AUC를 기준으로

top5 모델 선정!

Modeling

– Auto ML

1. 모델 생성 후 최적화

– gbc, lightgbm, catboost, lr, lda

```
# catboost
model_catboost = create_model('catboost', fold = 5)
model_catboost = tune_model(model_catboost, fold = 5, optimize = 'AUC', choose_better = True)
total_models.append(model_catboost)
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.6886	0.7589	0.7522	0.6324	0.6872	0.3818	0.3875
1	0.7034	0.7668	0.7692	0.6459	0.7022	0.4112	0.4174
2	0.6895	0.7621	0.7653	0.6305	0.6914	0.3848	0.3921
3	0.6955	0.7657	0.7774	0.6347	0.6988	0.3972	0.4056
4	0.6869	0.7554	0.7680	0.6269	0.6904	0.3801	0.3882
Mean	0.6928	0.7618	0.7664	0.6341	0.6940	0.3910	0.3982
Std	0.0060	0.0042	0.0081	0.0064	0.0056	0.0117	0.0116

gbc	0.761 -> 0.7597
lightgbm	0.759 -> 0.7601
catboost	0.758 -> 0.7631
lr	0.757 -> 0.7537
lda	0.756 -> 0.7544

Modeling

- Auto ML

2. 높은 성능을 보인 모델을 대상으로 bagging -> 성능 저하
- catboost, gbc, lightgbm

```
# bagging(n_estimators = 10 / 10번 복원 추출)
bag_catboost_10 = ensemble_model(model_catboost, n_estimators = 10, fold=5, optimize = 'AUC')
prediction_models.append(bag_catboost_10)
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.6905	0.7585	0.7609	0.6327	0.6909	0.3862	0.3928
1	0.7045	0.7672	0.7747	0.6459	0.7044	0.4138	0.4206
2	0.6895	0.7619	0.7701	0.6295	0.6927	0.3853	0.3933
3	0.6930	0.7647	0.7718	0.6331	0.6956	0.3919	0.3998
4	0.6861	0.7532	0.7660	0.6265	0.6892	0.3784	0.3863
Mean	0.6927	0.7611	0.7687	0.6335	0.6946	0.3911	0.3986
Std	0.0063	0.0049	0.0048	0.0066	0.0054	0.0121	0.0118

catboost	0.7631 -> 0.7611
gbc	0.7597 -> 0.7596
lightgbm	0.7601 -> 0.7584

3. blending을 통해 성능이 좋은 모델들을 합쳐 성능 개선 시도
- catboost + lightgbm + gbc : 0.7614

Modeling

– Auto ML

Catboost vs blending

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	CatBoost Classifier	0.6996	0.7711	0.7735	0.6369	0.6986	0.4048	0.4124

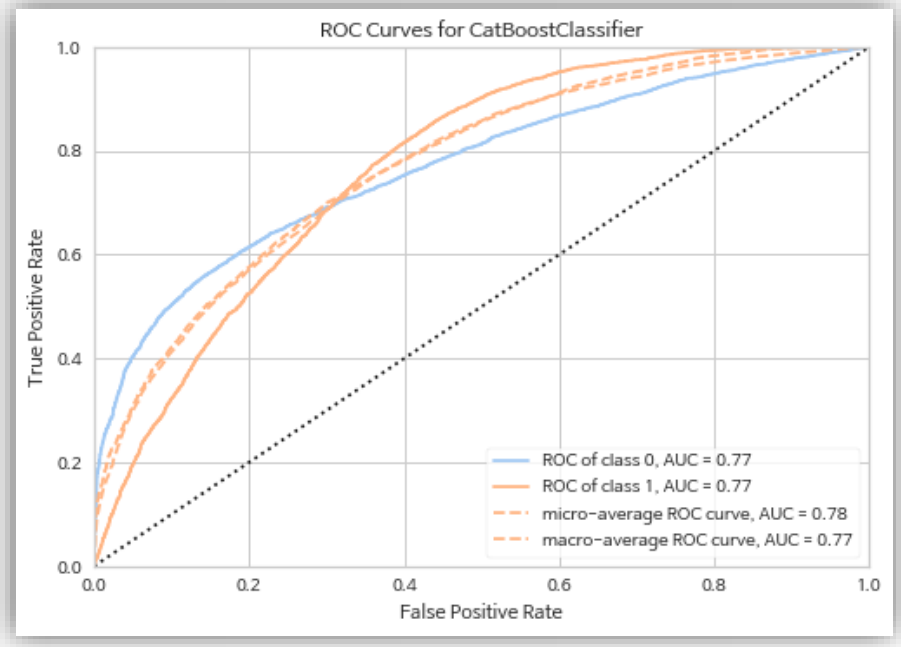
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Voting Classifier	0.6898	0.7633	0.6541	0.6596	0.6568	0.3739	0.3739

AUC 기준 catboost 성능이 더 좋음

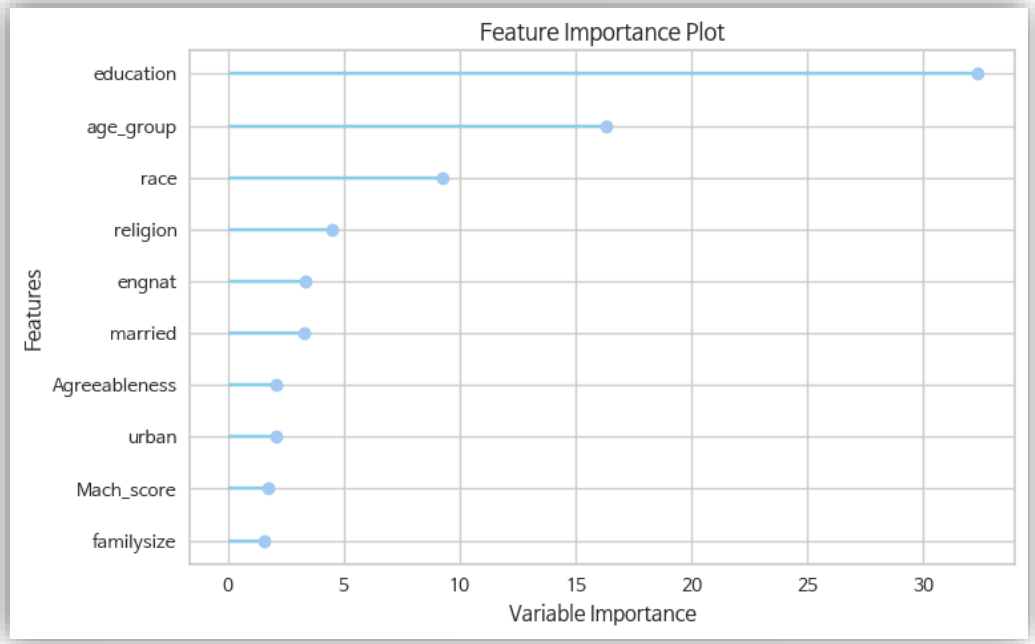
-> Catboost 하이퍼 파라미터 튜닝한 모델을 최종 모델로 선정!

Conclusion
-최종 결과

ROC Curves



Feature Importance Plot



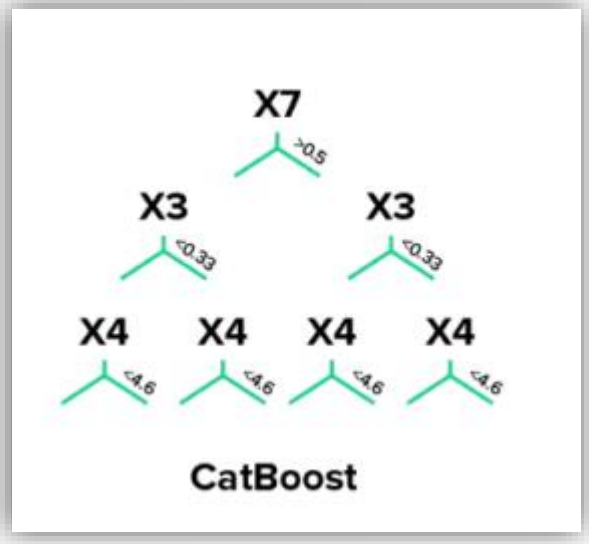
Conclusion
– 데이콘 제출 결과

721539	제출용 submission1.csv edit	2022-08-28 20:21:29	0.7035294123 0.703132614	

Conclusion

-최종 결과

catboost



- 잔차 추정의 분산을 최소로 하면서 bias 를 피하는 boosting 기법
- Ordered Boosting
- Ordered Target Encoding
- 범주형 변수에 대한 모델의 정확도와 속도가 높음
- 결측치가 많은 데이터셋에는 부적합

Conclusion

-한계 및 의의

1. Train set에서의 AUC가 높았지만 오버피팅 등의 문제로 데이콘 제출 결과 성능이 다소 감소하였다.

2. test set의 결측치 처리가 어려웠다.

3. Auto ML을 사용해 생각하지 못했던 모델들을 알고 성능을 높여 나갈 수 있었다.

4. 다양한 파생변수 생성 및 인코딩 방법을 시도해 보았다.

감사합니다