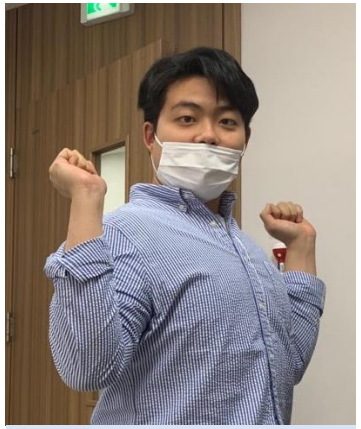
심리 성향 예측 AI 경진대회

ㅅㅅㅎㅎ

팀원 소개

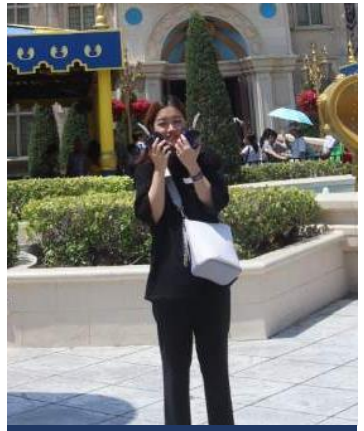


류시호

박동민

이상준

차노을



이혜린

심리성향 예측 ai 공모전

프로젝트명 : 심리성향 예측 ai 공모전

https://dacon.io/competitions/official/235647/overview/

프로젝트 내용 : 마키아벨리즘 심리테스트를 활용하여 테스트 참가자의 국가 선거 투표 여부 예측

개발 기간 : 2020.09 - 2020.10

사용언어 및 활용 기술 : 1) 파이썬 numpy, pandas 등의 패키지를 활용해 데이터를 가공, maltplot을 이용해 분석을 시각화

2) catboost, LGBM 모델등을 활용한 머신러닝

변수 설명

그림 출처 : https://m.blog.naver.com/PostView.nhn?blogId=nabisuper123&logNo=220745066511&proxyReferer=https:%2F%2Fwww.google.com%2F

마키아벨리즘 테스트란?

어둠의 3요소로 불리는 인격 특성 가운데 하나.

마키아벨리즘 성격의 구성개념은 16세기 정치가 Machiavelli의 실제 이야기에서 파생되었다.

성취동기가 높아 이를 충족시키기 위해 상대방을 쉽게 속이고, 사악하고 냉소적이며 교활하고 위선적인 성격 특성을 지니고 있다고 보고되었다.

조직내에서의 행동 관련 분야와 관련지어 연구되는 분야.

마키아벨리즘 성격특성을 가진 사람은 조직에 해가 되는 행동이나 반 생산적인 업무행동을 할 가능성이 높다.

타인에 대한 불신, 지위에 대한 욕구, 통제에 대한 욕구, 비도덕적 조종의 4가지 하위 요인으로 구성

류시호	차노을	박동민	이혜린	이상준
73	72	68	67	58

마키아벨리즘 테스트 문항

대표문항

13. 나는 타인을 신뢰하지 않기 때문에, 집단에 참여하는 것을 좋아하지 않는다.

15. 내가 일을 수행하는데 어떤 약점을 보인다면, 다른 사람들이 그것을 이용할 것이다.

12. 사람들은 오직 개인적인 이득에 의해서 동기유발 된다

1. 내가 성공하는데 도움이 된다고 믿는다면 나는 비윤리적인 것이라도 기꺼이 할 것이다.

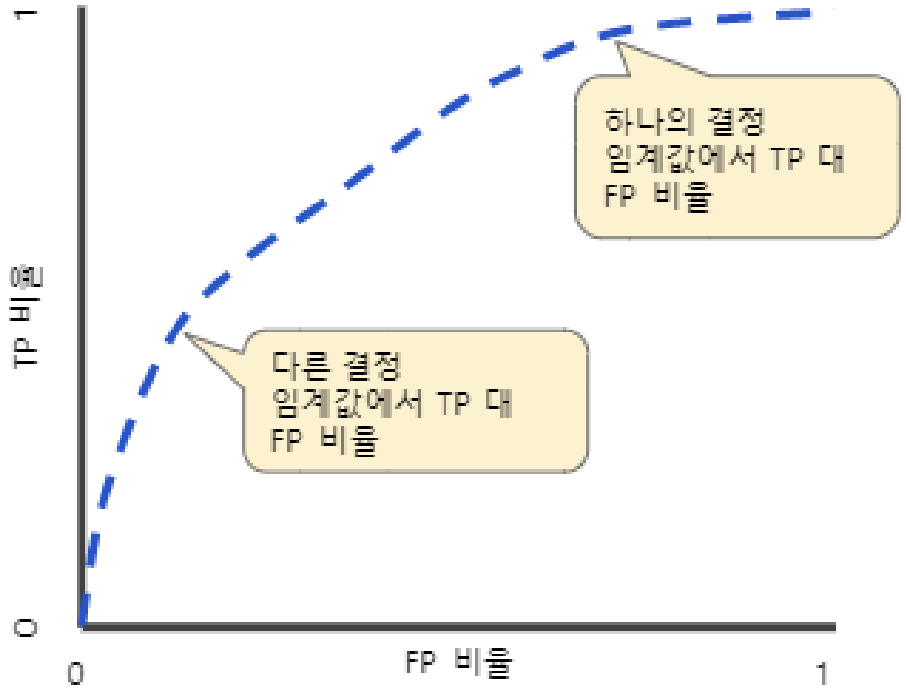
-
-
-

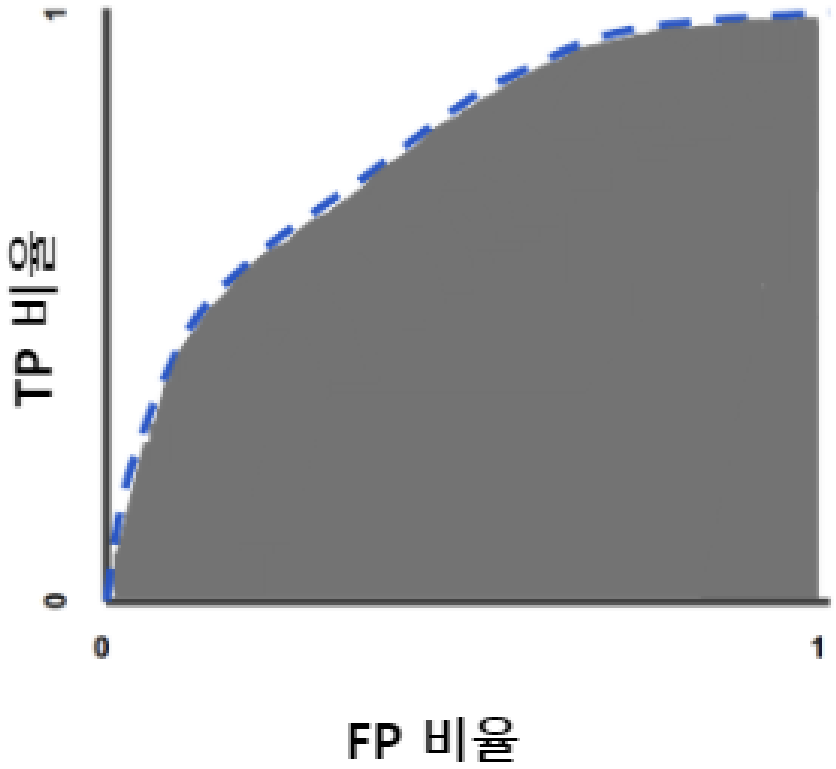
출처 : 한국판 마키아벨리즘 성격 척도(MPS)의 타당화 및 신뢰도 연구

변수 설명

Q_A	테스트 문항
Q_E	테스트 질문까지 걸린 시간
age_group	연령
education	교육수준
engnat	모국어가 영어인가 아닌가
Familysize	형제, 자매 수
gender	성별
hand	오른손잡이, 왼손잡이, 양손잡이
married	결혼유무
race	인종
religion	종교
tp_	본인이 생각하는 성향
urban	고향(유년기에 살았던 곳)
voted	전년도 투표 여부
wf_	허구인 단어의 의미를 안다
wr_	실존하는 단어의 의미를 안다

평가지표 : AUC





$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=ko

전처리

Married

미혼, 이혼, 기혼 이외의 값(other, 93ea)은 미혼 값으로 포함

age_group

10대 그룹의 투표여부가 현저하게 차이가 나므로 10대 그룹과 그 외로 나눔

생애주기 이론을 통해 미성년(~19)/성년(20~39)/중년(40~59)/노년(60~)으로 구분

race

인원수가 많은 인종(White, Black, Asian)을 제외한 나머지 인종 그룹화

Q_E 그룹

변수 간의 편차가 크다. (최대값 : 10706013, 최소값 : 0)

절대적인 시간 변수가 아닌 상대적인 변수 이므로 일정한 기준을 통해 분류

quantile 함수를 사용해 각각 25% 값과 75% 값으로 나누어 그룹화

education

무응답 값을 Less than high school로 추가

Wr_new

실존하는 단어들의 의미를 아는가에 대한 질문에 4가지 질문만 모른다의 답변이 월등하게 높으므로 따로 그룹화 해서 확인

Wf_new

허구의 단어들의 의미를 하나라도 안다고 체크한 경우를 새로운 컬럼으로 생성하여 투표여부와 비교

신뢰도와 상관이 있다고 가정

tp

각 tp를 그룹화 하여 대표적인 성향을 분류

Race_white

백인과 백인이 아닌 사람들을 구분했을 때가 상관계수가 높았기 때문에 변수로 추가.

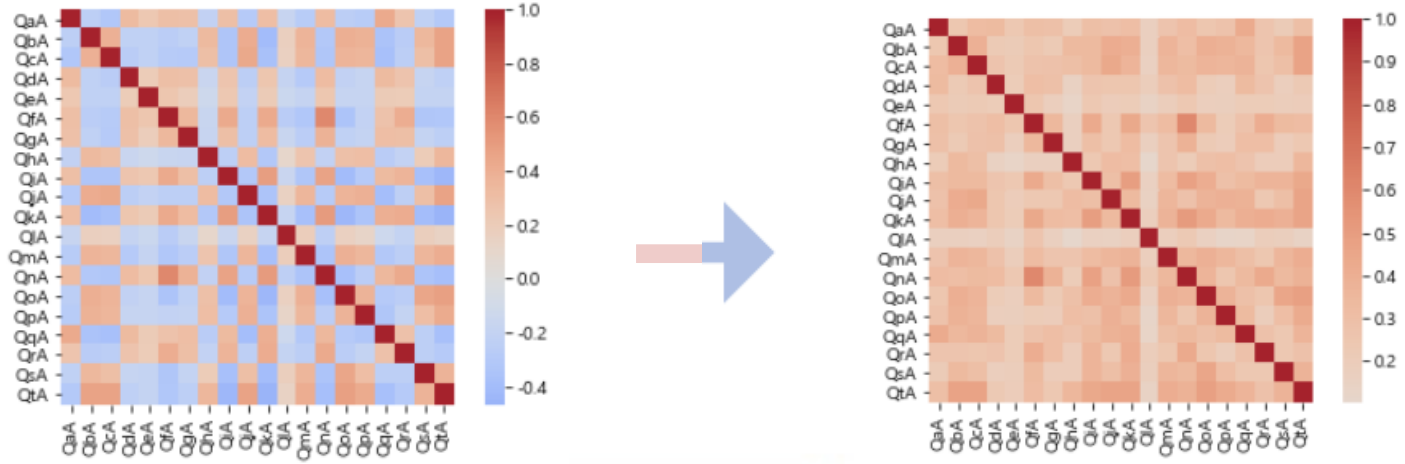
Mach_score

마키아벨리즘 테스트를 점수화하여 투표여부와 상관관계를 확인할 필요성이 있다.

역문항을 구별해 내기 위하여 히트맵을 통해 문항 간의 상관계수를 구한다.

문항 간의 상관계수를 토대로 역문항을 구별해낸다.

역문항을 reverse 처리한 후 전체문항을 더해 평균으로 나눈다.



머신러닝

그림 출처 : http://www.edupolnews.com/news/articleView.html?idxno=11840

Pycaret 라이브러리를 활용한 AutoML

Pycaret 공식 홈페이지 : https://pycaret.org/

파이썬 라이브러리 pycaret

간단한 모델링, 하이퍼 파라미터 튜닝, feature importance 등 작업을 한번에!

AutoML 공식 사이트 : https://cloud.google.com/automl?hl=ko

```
from pycaret.classification import *

# 'voted' 컬럼이 예측 대상이므로 target 인자에 명시
# 'voted' column is the target variable
clf = setup(data = train, target = 'voted')
```

Setup Succesfully Completed!

	Description	Value
0	session_id	1984
1	Target Type	Binary
2	Label Encoded	0: 0, 1: 1
3	Original Data	(45532, 139)
4	Missing Values	False
5	Numeric Features	48
6	Categorical Features	90
7	Ordinal Features	False
8	High Cardinality Features	False
9	High Cardinality Method	None
10	Sampled Data	(45532, 139)
11	Transformed Train Set	(31872, 238)
12	Transformed Test Set	(13660, 238)
13	Numeric Imputer	mean
14	Categorical Imputer	constant
15	Normalize	False
16	Normalize Method	None
17	Transformation	False

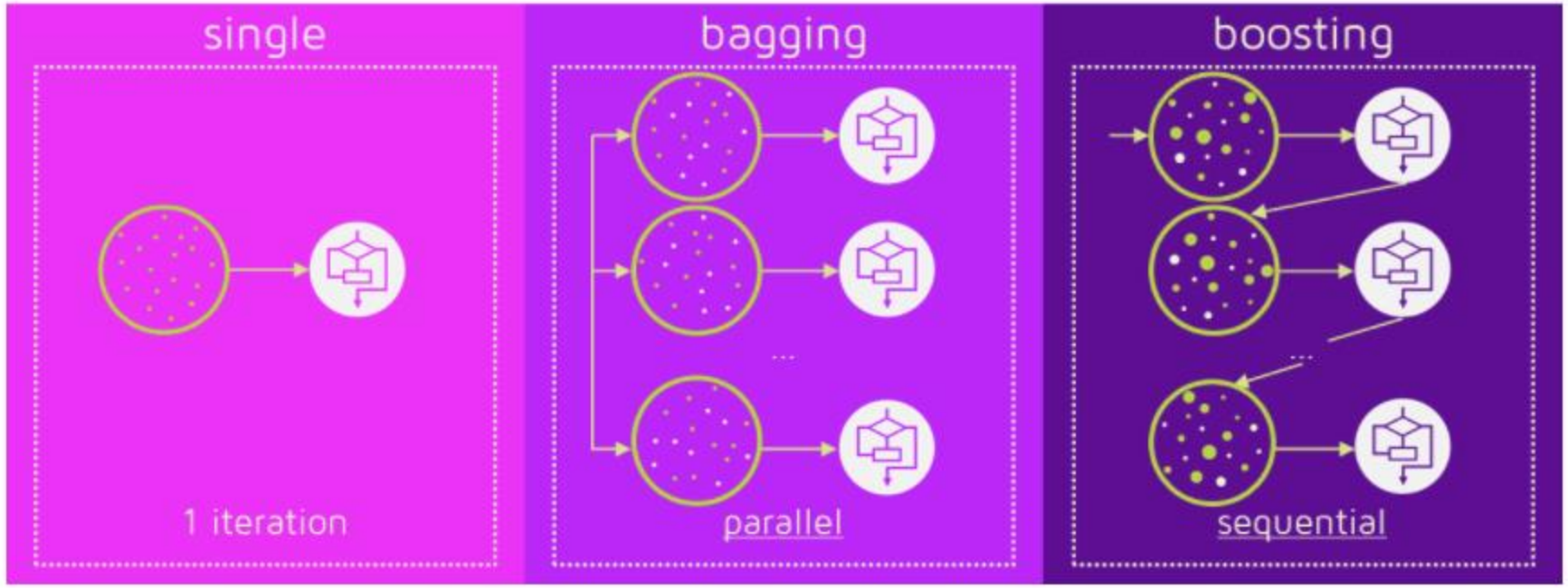
Pycaret 라이브러리를 활용한 AutoML

머신러닝으로 모델링할 때 사용되는
대부분의 알고리즘들은 다 구성되어 있고,
이들 중 어떤 모델이 가장 성능이 좋은지 확인 가능

```
best_3 = compare_models(sort = 'AUC', n_select = 3)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	CatBoost Classifier	0.6927	0.7649	0.6564	0.7504	0.7002	0.3879	0.3916	34.5020
1	Gradient Boosting Classifier	0.6934	0.7634	0.6412	0.7606	0.6957	0.3912	0.3971	60.9425
2	Light Gradient Boosting Machine	0.6924	0.7630	0.6451	0.7567	0.6963	0.3887	0.3939	1.3934
3	Extra Trees Classifier	0.6886	0.7601	0.6459	0.7500	0.6940	0.3806	0.3851	5.5389
4	Linear Discriminant Analysis	0.6879	0.7585	0.6574	0.7424	0.6973	0.3776	0.3807	1.4200
5	Ada Boost Classifier	0.6845	0.7535	0.6482	0.7423	0.6920	0.3717	0.3754	13.6749
6	Extreme Gradient Boosting	0.6755	0.7449	0.6639	0.7207	0.6911	0.3506	0.3519	7.7193
7	Random Forest Classifier	0.6596	0.7150	0.6096	0.7243	0.6620	0.3243	0.3293	0.7514
8	Decision Tree Classifier	0.6102	0.6066	0.6443	0.6433	0.6437	0.2133	0.2133	4.2429
9	Naive Bayes	0.4539	0.5496	0.0138	0.5211	0.0268	-0.0011	-0.0057	0.2148
10	Quadratic Discriminant Analysis	0.5231	0.5225	0.6223	0.5585	0.5801	0.0258	0.0272	0.4669
11	K Neighbors Classifier	0.5140	0.5128	0.5559	0.5556	0.5557	0.0193	0.0193	3.6718
12	Logistic Regression	0.5469	0.4831	0.9984	0.5469	0.7067	0.0006	0.0043	1.3287
13	SVM - Linear Kernel	0.4990	0.0000	0.5150	0.5445	0.5266	-0.0054	-0.0055	0.5351
14	Ridge Classifier	0.6879	0.0000	0.6578	0.7423	0.6974	0.3777	0.3807	0.2504

Boosting



Boosting 은 다른 기법과는 달리 앙상블을 반복하는 과정에서 이전 과정의 결과가 다음 과정의 결과에 영향을 미침

오답에 대해서 높은 가중치를 부여하여 집중적으로 이를 고치게함으로써 모델의 정확도를 크게 향상시킴

모델 앙상블 (Model Ensemble)

여러 개의 모델을 결합하여 하나의 모델 보다 더 좋은 성능을 내는 머신 러닝 기법

blend_models

블렌딩의 개념은 다양한 기계 학습 알고리즘을 결합하고 분류의 경우 최종 결과를 예측하기 위해 과반수 투표 또는 평균 예측 확률을 사용하는 것입니다.

```
blended1 = blend_models(estimator_list = best_3, fold = 5, method = 'soft')
```

'소프트 '또는 ' 하드 ' 를 정의 할 수 있습니다.

'소프트는 투표에 예측 확률을 사용하고 하드는 예측 레이블을 사용합니다

모델 앙상블 (Model Ensemble)

모델 튜닝

모델을 하이퍼파라미터로 자동으로 튜닝시켜주는 기능

```
tuned_cat = tune_model(cat, optimize = 'AUC', n_iter = 10)
tuned_gb = tune_model(gb, optimize = 'AUC', n_iter = 10)
tuned_lgbm = tune_model(lgbm, optimize = 'AUC', n_iter = 10)
```

```
tune_blended = blend_models(estimator_list = ['tuned_cat','tuned_gb','tuned_lgbm'], fo
```

모델 예측 (Prediction)

```
pred_holdout = predict_model(blended1)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Voting Classifier	0.6908	0.7578	0.643	0.7553	0.6946	0.3857	0.3908

```
pred_holdout = predict_model(tune_blended)
```

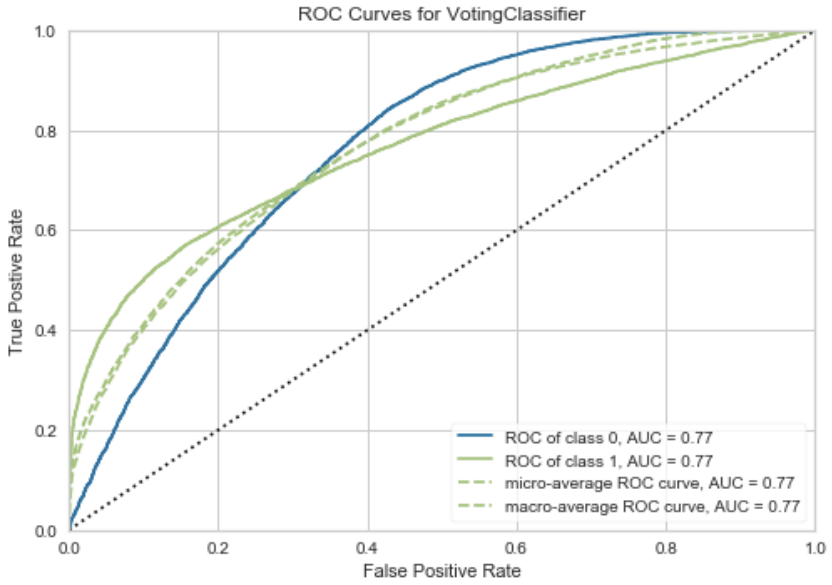
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Voting Classifier	0.6916	0.7658	0.6556	0.7491	0.6992	0.3856	0.3893

모델 앙상블 (Model Ensemble)

TEST 결과 예측 (Prediction)

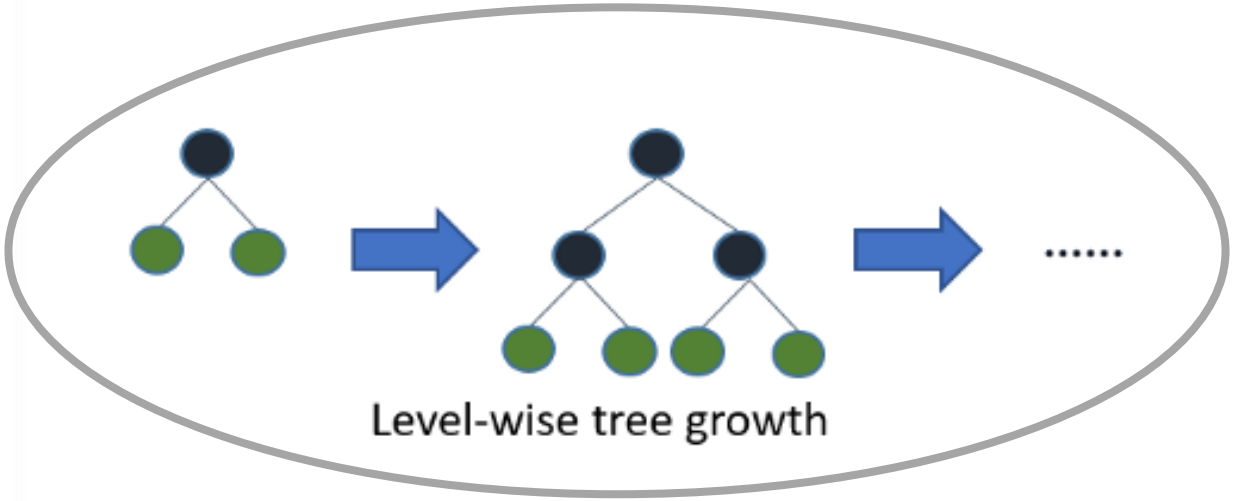
```
predictions_tune = predict_model(final_model_tune, data = test)
```

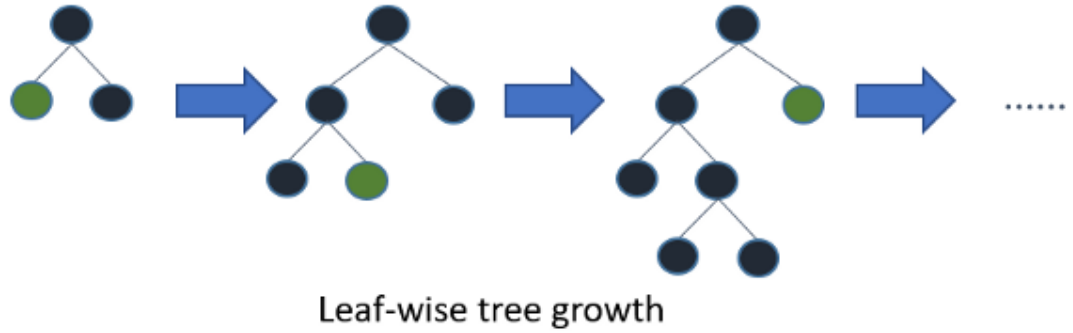
```
predictions_tune['Score']
```



466588 **sample_submission_ens_final_tune.csv** 2020-10-21 14:57:29 0.7780925745

Catboost





범주형 피쳐를 지원하는 Gradient Boosting 오픈소스 라이브러리로서 기본 예측기로는 의사결정 나무 사용

기존 부스팅 모델의 느린 학습속도와 오버피팅 문제 해결이 가능한 모델

대부분의 의사 결정 트리 학습 알고리즘은 다음 이미지와 같이 수준 (깊이) 별로 트리를 확장합니다.

공식 사이트 : https://catboost.a

Catboost

기존의 부스팅 기법은 모든 데이터(x1 ~ x10) 까지의 잔차를 일괄 계산한다.

time	datapoint	class label
12:00	x1	10
12:01	x2	12
12:02	x3	9
12:03	x4	4
12:04	x5	52
12:05	x6	22
12:06	x7	33
12:07	x8	34
12:08	x9	32
12:09	x10	12

Catboost 잔차 계산법

1. 먼저 x1 의 잔차만 계산하고, 이를 기반으로 모델을 만든다. 그리고 x2 의 잔차를 이 모델로 예측한다.

2. x1, x2 의 잔차를 가지고 모델을 만든다. 이를 기반으로 x3, x4 의 잔차를 모델로 예측한다.

3. x1, x2, x3, x4 를 가지고 모델을 만든다. 이를 기반으로 x5, x6, z7, x8 의 잔차를 모델로 예측한다.

4. ... 반복

이 과정에서 데이터 순서를 셔플링하는 등의 방법으로 트리를 다각적으로 만들어 오버피팅을 방지합니다

출처: https://dailyheumsi.tistory.com/136 [하나씩 점을 찍어 나가며]

Catboost

Catboost 모델링

```python
from catboost import CatBoostClassifier

clf = CatBoostClassifier(
    iterations=5,
    learning_rate=0.1,
    #loss_function='CrossEntropy'
)


clf.fit(X_train, y_train,
#         cat_features=cat_features,
#         eval_set=(X_val, y_val),
#         verbose=False
)

print('CatBoost model is fitted: ' + str(clf.is_fitted()))
print('CatBoost model parameters:')
print(clf.get_params())
```

```
0:      learn: 0.6621402        total: 67.9ms   remaining: 272ms
1:      learn: 0.6446062        total: 80.8ms   remaining: 121ms
2:      learn: 0.6306544        total: 90ms     remaining: 60ms
3:      learn: 0.6195858        total: 99.4ms   remaining: 24.9ms
4:      learn: 0.6066691        total: 109ms    remaining: 0us
CatBoost model is fitted: True
CatBoost model parameters:
{'iterations': 5, 'learning_rate': 0.1}
```

Catboost

Test결과 예측

```
y_pred_pro1 = clf.predict_proba(test)
```

```

y_pred_pro11=[]

for i in range(len(y_pred_pro1)) :
    y_pred_pro11.append(y_pred_pro1[i][1])

y_pred_pro11
```

```
[0.6342984078512366,
 0.708097084284256,
 0.5123908697057471,
 0.4241488830434792,
 0.6409685388010583,
 0.5070768318970229,
 0.7059251203615223,
 0.5095393443721551,
 0.4420515215016714,
 0.5275801017488848,
 0.4305896005703282,
 0.6184903503064688,
 0.5108169313408684,
 0.4872592243080558,
 0.5107431440763281,
 0.7515244497413304,
 0.7153370691747674,
 0.4291026738857129,
 0.43655087412696647,
```

```
submission['voted']=y_pred_pro11
submission=submission.reset_index()
```

```
submission  ##결과값 12인지 확인
```

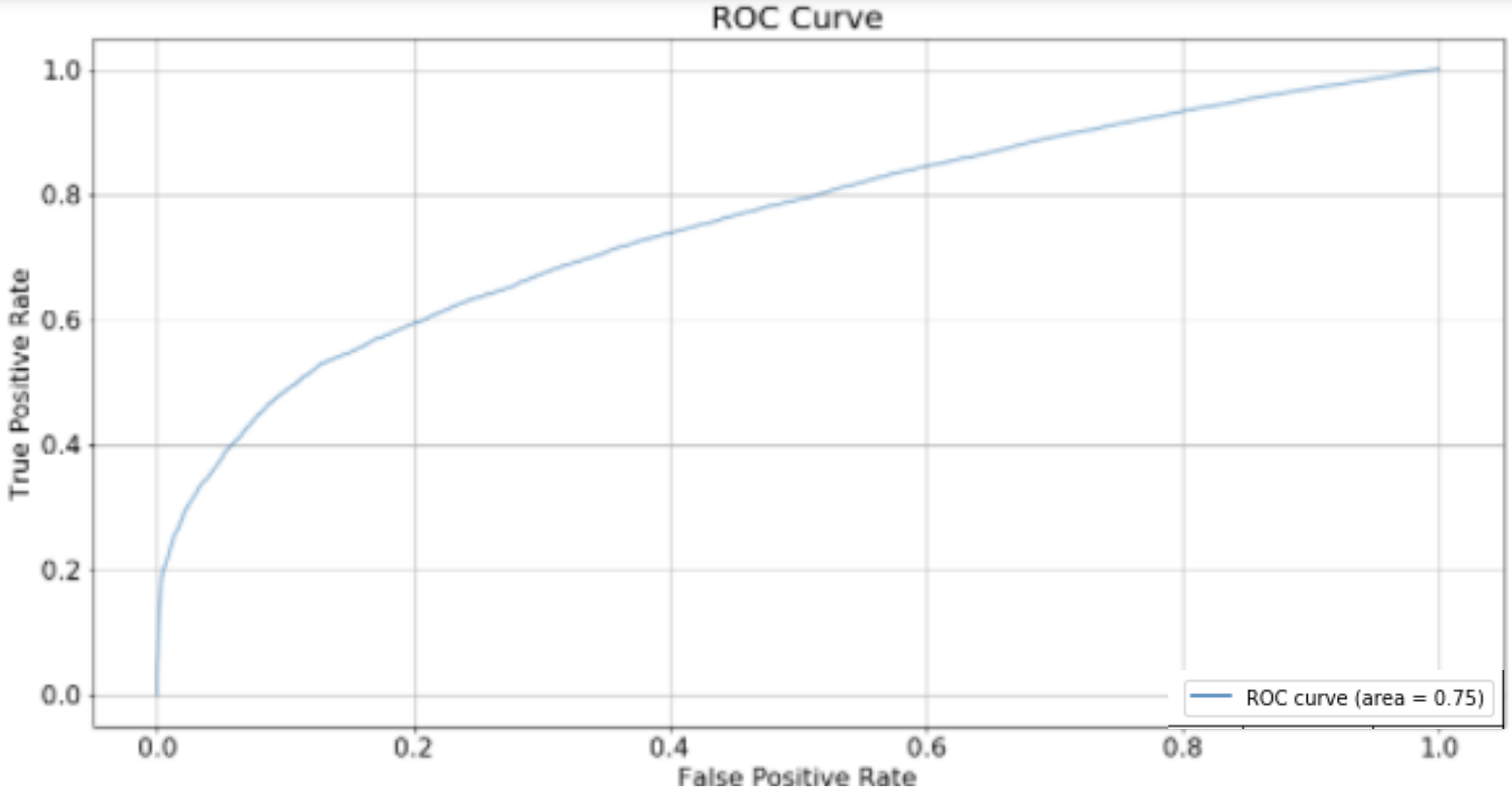
	index	voted
0	0	0.634298
1	1	0.708097
2	2	0.512391
3	3	0.424149
4	4	0.640969
...	...	...
11378	11378	0.483723
11379	11379	0.717452
11380	11380	0.430553
11381	11381	0.464046
11382	11382	0.545506

11383 rows × 2 columns

```
submission.to_csv('C:₩₩Users₩₩acorn₩₩Desktop₩₩sample_submission_cat_finalpro.csv'
```

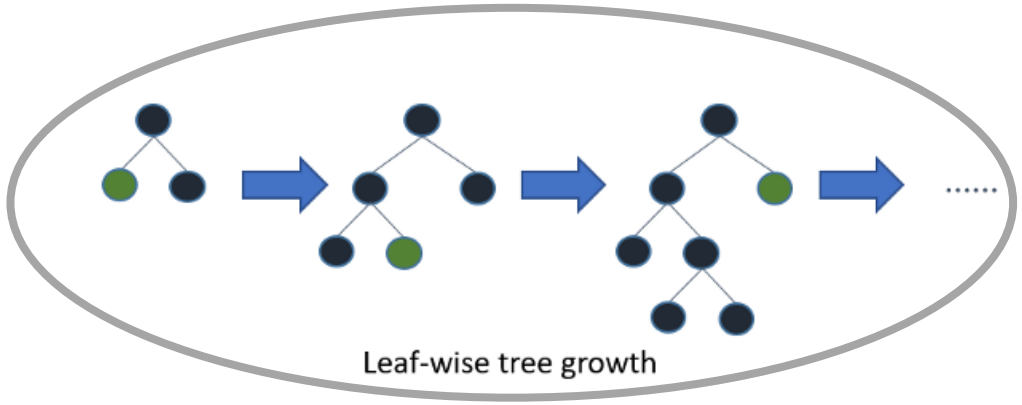
Catboost

Test결과 예측



466514 **sample_submission_cat_final.csv** 2020-10-21 14:57:29 0.75727

LGBM(Light Gradient Boosting Machine)



Light GBM은 트리 기반의 학습 알고리즘인 gradient boosting 방식의 프레임 워크이다.

다른 알고리즘은 나무를 수평으로 확장하는 반면, LGBM은 나무를 수직으로 확장한다,

다른 알고리즘보다 낮은 loss를 달성하는 경향이 있다.

최대 손실 값을 가지는 리프노드를 지속적으로 분할하면서 트리가 깊어지고 비대칭적으로 생성하여 예측 오류 손실을 최소화

LGBM 공식 홈페이지 : https://lightgbm.readthedocs.io/en/latest/#

LGBM(Light Gradient Boosting Machine)

장점	XGBoost 대비 더 빠른 학습과 예측 수행 시간 더 작은 메모리 사용량 카테고리형 피처의 자동 변환과 최적 분할 높은 모델의 해석력 이상치와 노이즈에 큰 영향을 받지 않음
단점	적은 데이터 세트에 적용할 경우 과적합이 발생하기 쉬움

Hyperparameter Optimization

```
lgbmBO = BayesianOptimization(f = lgbm_cv,
                              pbounds = {'learning_rate': (0.0001, 0.1),
                                         'colsample_bytree': (0, 1),
                                         'num_leaves': (16, 1024),
                                         'max_depth': (2, 25),
                                         'min_child_weight': (30, 100),
                                         'feature_fraction': (0.0001, 0.99),
                                         'bagging_fraction': (0.0001, 0.99),
                                         'lambda_l1' : (0, 0.99),
                                         'lambda_l2' : (0, 0.99),
                                         'n_estimators': (16, 1024),
                                         'subsample': (0, 1),
                                         'reg_alpha': (0, 10),
                                         'reg_lambda': (0, 50)
                                        },
                              verbose = 2,
                              random_state = 0 )
```

학습을 수행하기 위해 사전에 설정해야 하는 값인
파라미터의 최적값을 탐색하는 문제

파라미터 명 (파이썬 래퍼)	파라미터명 (사이킷런 래퍼)	설명
num_iterations (100)	**n_estimators** (100)	- 반복 수행 트리개수 지정 - 너무 크면 과적합 발생
learning_rate (0.1)	**learning_rate** (0.1)	- 학습률
max_depth (-1)	**max_depth** (-1)	- 최대 깊이 - default → 깊이에 제한이 없음
min_data_in_leaf (20)	**min_child_samples** (20)	- 최종 리프 노드가 되기 위한 레코드수 - 과적합 제어용
num_leaves (31)	**num_leaves** (31)	- 하나의 트리가 가지는 최대 리프 개수
boosting ('gbdt')	**boosting_type** ('gbdt')	- gbdt : 일반적인 그래디언트부스팅 트리 - rf : 랜덤포레스트
bagging_fraction (1.0)	**subsample** (1.0)	- 데이터 샘플링 비율 - 과적합 제어용
feature_fraction (1.0)	**colsample_bytree** (1.0)	- 개별트리 학습시 선택되는 피처 비율 - 과적합 제어용
lambda_l2 (0)	**reg_lambda** (0)	- L2 Regularization 적용 값 - 피처 개수가 많을 때 적용을 검토 - 클수록 과적합 감소 효과
lambda_l1 (0)	**reg_alpha** (0)	- L1 Regularization 적용 값 - 피처 개수가 많을 때 적용을 검토 - 클수록 과적합 감소 효과
objective	**objective**	- 'reg:linear' : 회귀 - binary:logistic : 이진분류 - multi:softmax : 다중분류, 클래스 반환 - multi:softprob : 다중분류, 확률반환

Bayesian Optimization

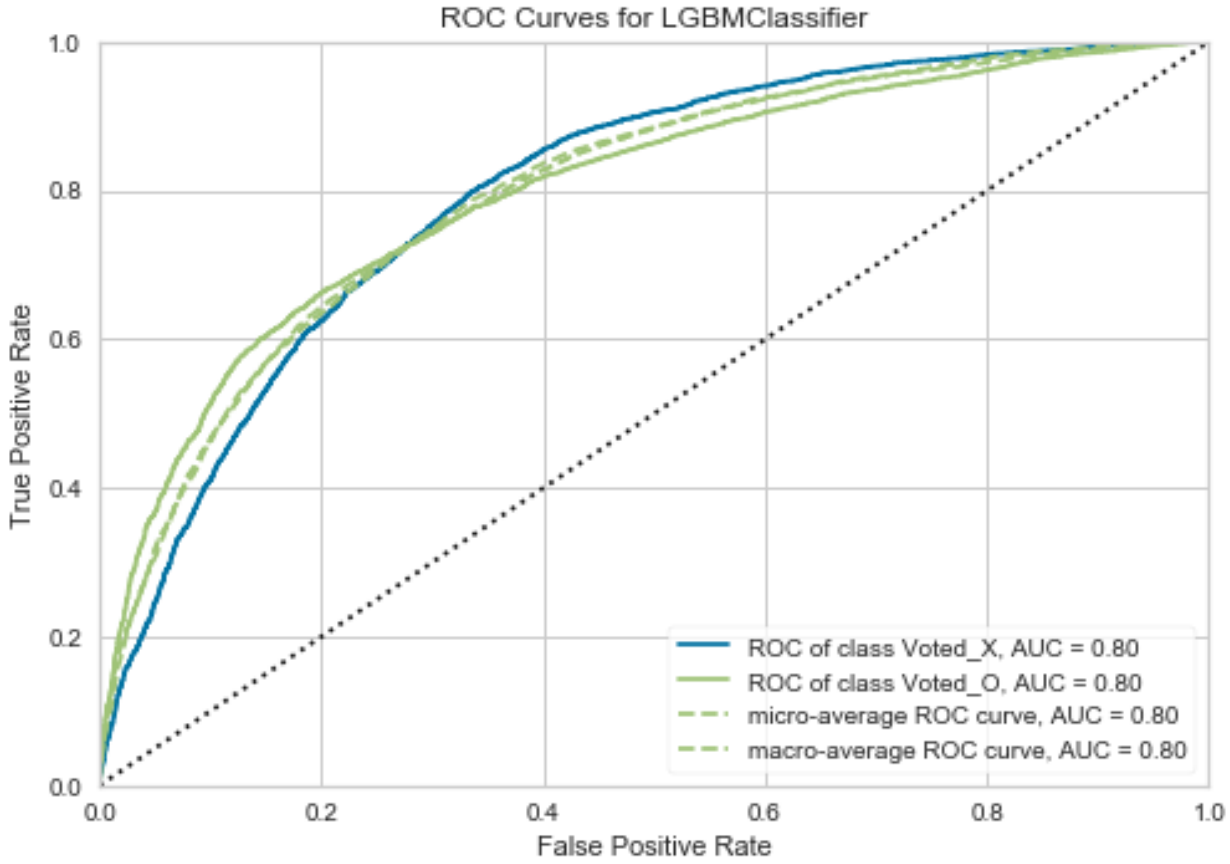
매 회 새로운 hyperparameter 값에 대한 조사를 수행할 시 '사전 지식'을 충분히 반영하면서, 동시에 전체적인 탐색 과정을 좀 더 체계적으로 수행하기 위해 고려해볼 수 있는 Hyperparameter Optimization 방법론

iter	target	colsam...	learni...	max_depth	min_ch...	n_esti...	num_le...
1	0.6872	0.5488	0.07155	15.86	68.14	443.0	667.1
2	0.6727	0.3834	0.07919	14.16	69.76	949.0	87.6
3	0.6826	0.7782	0.08701	24.51	85.94	481.2	802.8
4	0.6821	0.9447	0.05223	11.54	48.52	796.4	475.8
5	0.6901	0.6121	0.06173	23.71	77.73	378.4	456.5
6	0.6546	0.5004	0.01961	14.12	37.3	21.58	502.4
7	0.6803	0.004703	0.09826	19.85	53.59	805.1	468.9
8	0.6966	0.3789	0.05745	2.6	50.19	559.7	511.5
9	0.6943	0.8981	0.08173	2.428	33.26	531.9	257.1
10	0.6924	0.9042	0.08041	9.741	91.29	380.6	23.01
11	0.7012	0.5858	0.006859	14.27	60.32	443.6	657.7
12	0.6944	0.1215	0.02012	18.19	73.68	435.7	649.8
13	0.687	0.7149	0.07061	13.88	76.45	436.7	649.1
14	0.6923	0.5809	0.05752	8.554	45.61	452.4	646.5
15	0.6875	0.7254	0.08547	15.45	94.08	381.6	33.21
16	0.6945	0.4579	0.05414	5.823	37.54	537.3	257.1
17	0.6935	0.8661	0.04833	8.691	63.23	440.8	657.7
18	0.7	0.924	0.00684	22.23	62.55	456.0	640.5
19	0.6828	0.9288	0.06536	12.75	34.49	621.8	480.0
20	0.688	0.1535	0.08866	23.29	34.66	432.7	647.6
21	0.6892	0.3175	0.04914	11.17	63.46	998.4	333.0
22	0.7001	0.9965	0.04985	24.2	89.78	93.14	318.4
23	0.6895	0.9414	0.029	18.97	43.33	774.9	376.5
24	0.7007	0.6647	0.008791	21.65	63.46	465.4	905.0
25	0.6898	0.589	0.04448	13.46	30.36	607.1	887.8

LGBM을 위한 전처리

encoding	**one-hot** bias가 가장 적다. **frequency** 값 분포에 대한 정보가 잘 보존 값의 빈도가 타겟과 연관이 있으면 유용 **mean(smooth)** 타겟을 이용하여 평균값을 얻어내 설명
scaling	**minmax** 최소값(min)과 최대값(max)을 사용하여 '0~1' 사이의 범위로 데이터를 표준화

결과



466426 **MINMAX_X.csv**

2020-10-20 17:04:23 0.7728092808

느낀 점 및 한계점

그림 출처 : https://m.blog.naver.com/PostView.nhn?blogId=xiaoeun1205&logNo=220511063250&proxyReferer=https%3A%2F%2Fwww.google.com%2F

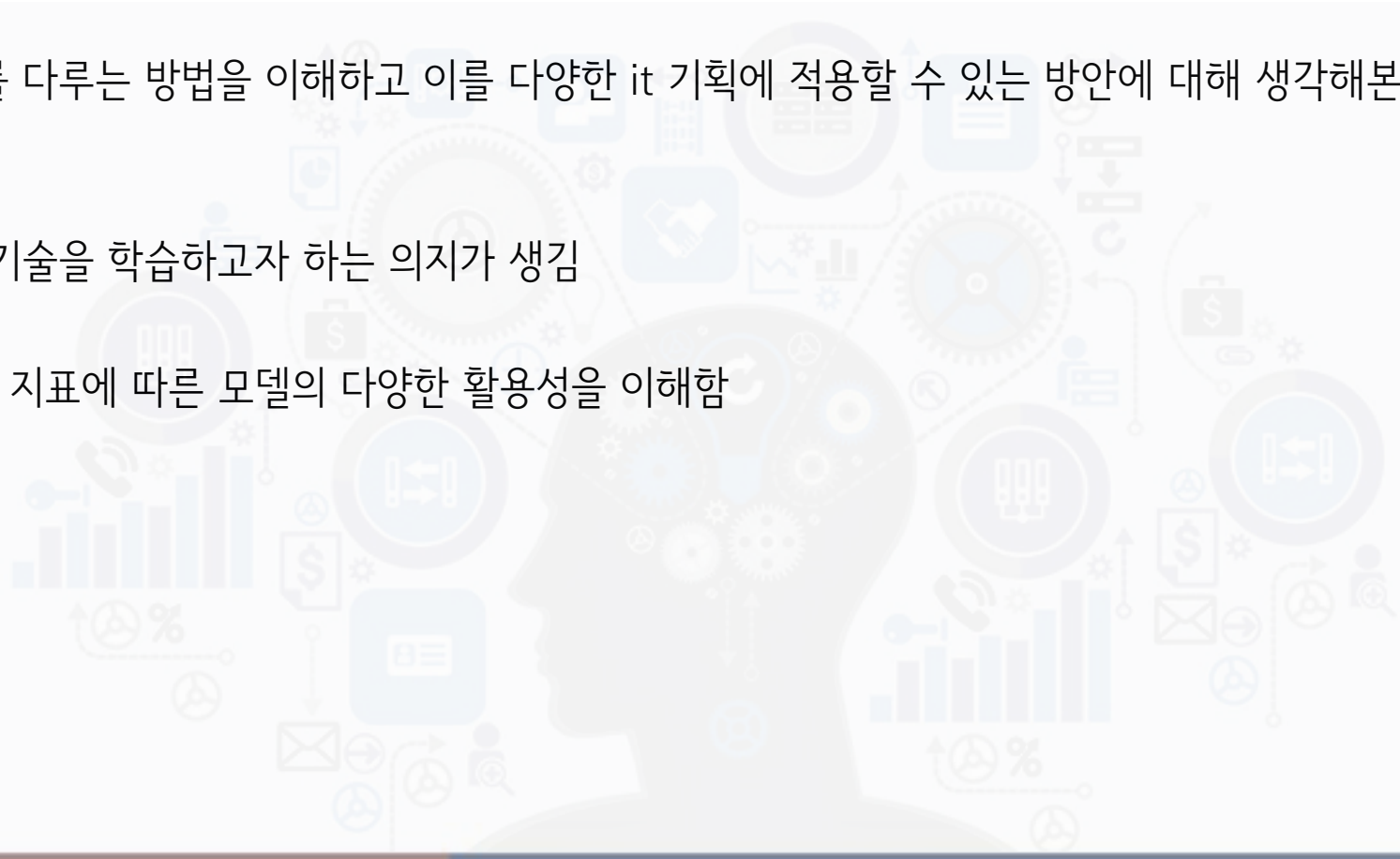
느낀 점

심리학 테스트의 범주가 넓어짐에 따라 해당 영역의 데이터 분석 방법 탐구함

심리 설문조사(범주형 변수)를 다루는 방법을 이해하고 이를 다양한 it 기획에 적용할 수 있는 방안에 대해 생각해본 계기가 됨

빠르게 변하는 분석 트렌드, 기술을 학습하고자 하는 의지가 생김

파라미터 최적화를 통해 평가 지표에 따른 모델의 다양한 활용성을 이해함

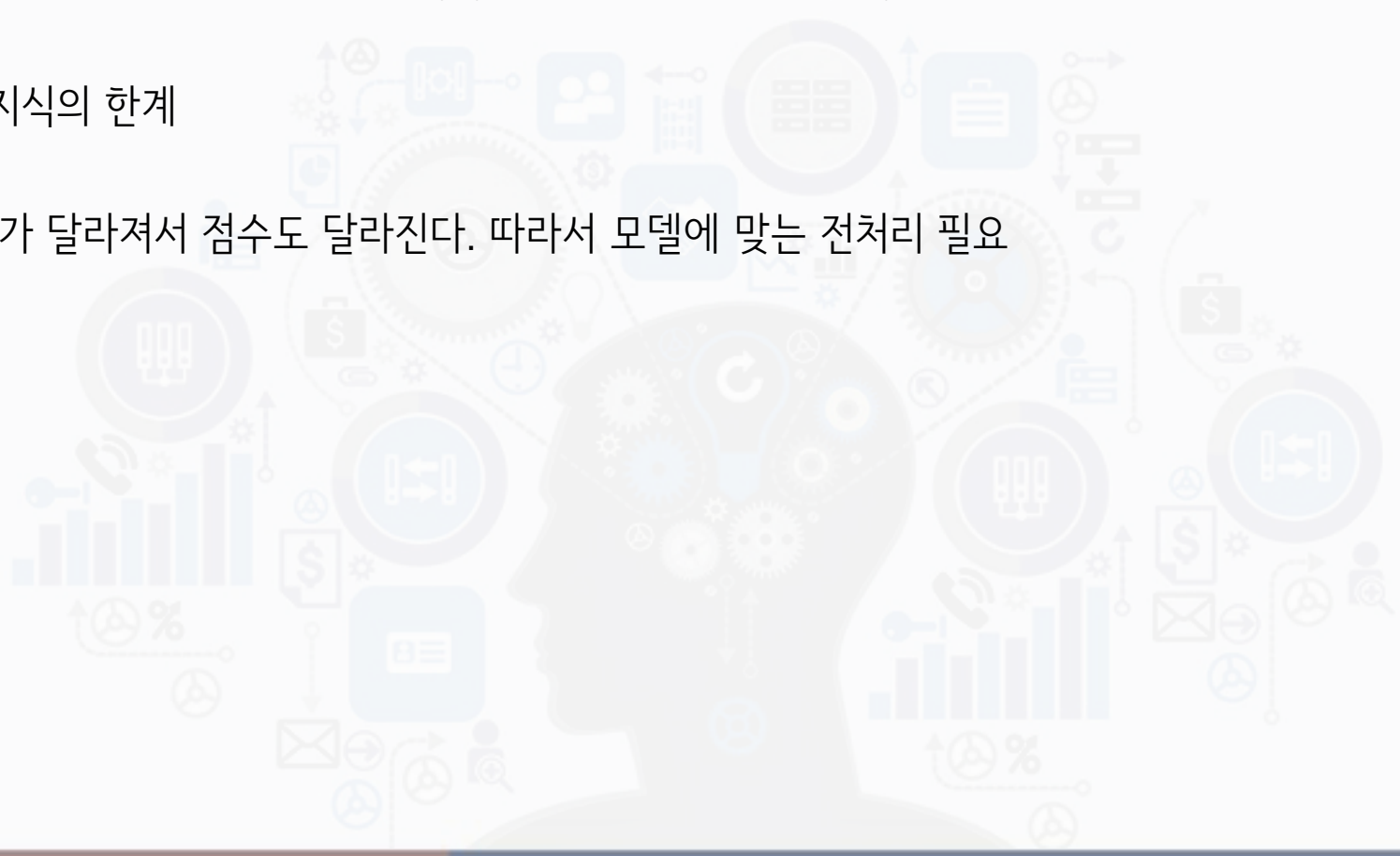


한계점

데이터 자체의 수 부족, 마키아벨리즘 테스트 자체의 신빙성 부족 및 투표여부와의 상관관계 부족

머신 러닝 모델에 대한 기본 지식의 한계

전처리에 따라 모델의 적합도가 달라져서 점수도 달라진다. 따라서 모델에 맞는 전처리 필요



감사합니다

https://github.com/Chachaeul/TeamProject