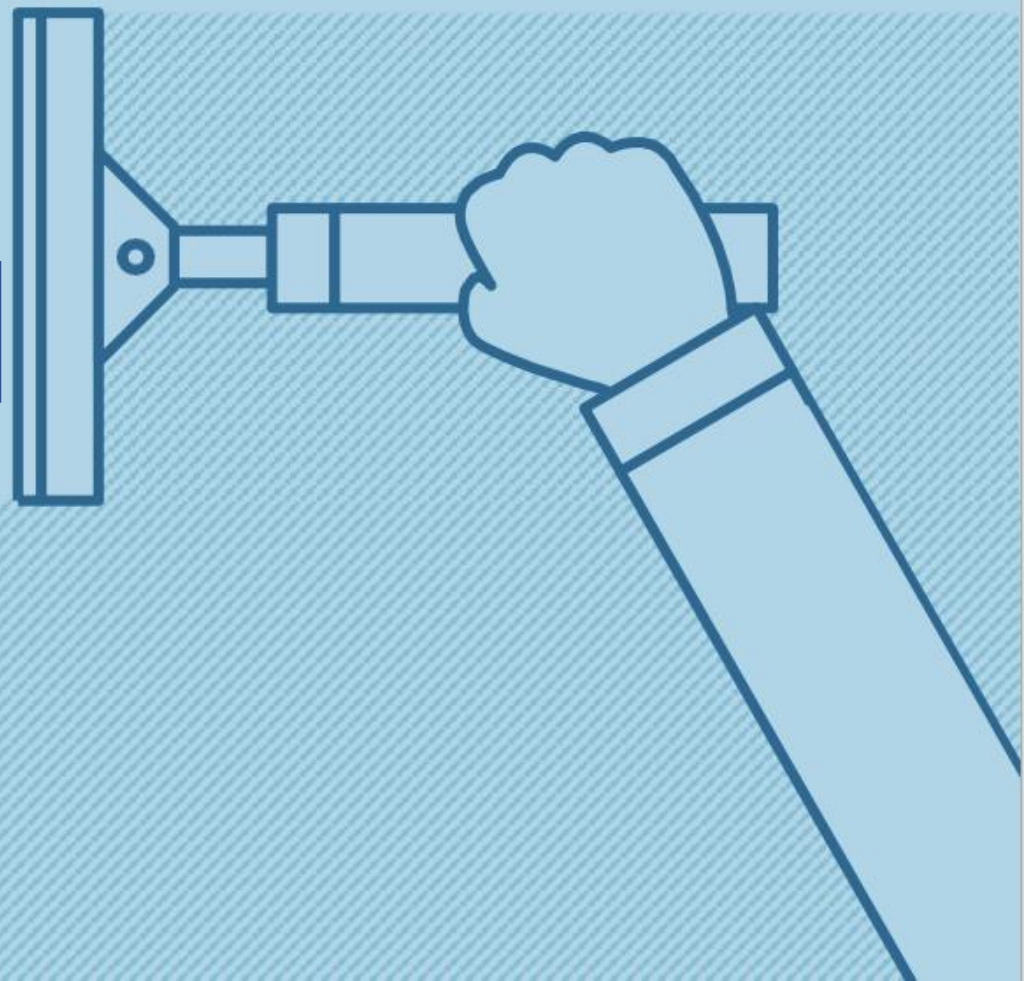
Game bot detection

R모르 파티
김정학 백진우 정희영 최종문



CONTENTS

1

EDA

Log를 통해 column 생성 및 재구성

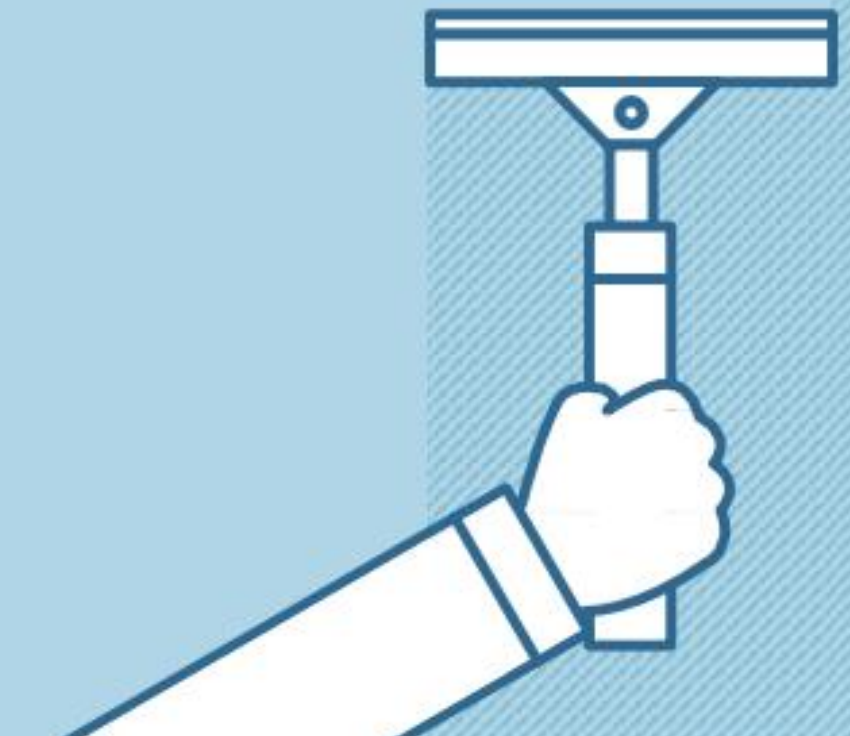
2

사용 모델 및 성능평가

사용 모델 선정 및 성능

Column 생성

Log를 통해 새로운 column 생성



1. Column 생성

Column 종류

	의미	비고
Player information	플레이어의 정보(로그인, 게임시간, 돈, 레벨)	위의 3항목은 논문과 비슷하게 구성하였고 Network를 새롭게 추가해주었다.
Player action	플레이어의 행동(앉기, 경험치&돈 획득, 포탈 사용, Kill, **퀘스트**)	
Group activity	플레이어의 그룹 활동(파티, 길드)	
Network activity	플레이어의 네트워킹(친구, 거래, 우편)	

1. Column 생성

Player information

Column	사용된 로그 정보
login_count	로그인 횟수
logout_count	로그아웃 횟수
login_day_count	당일 로그인 여부
play_time	플레이 시간
avg_money	평균 돈 소지량
ip_count	ip의 개수
max_level	최고 레벨

1. Column 생성

Player action

Column	사용된 로그 정보
sit_count	앉은 횟수
exp_get_amount	경험치 획득량
item_get_count	아이템 획득량
exp_repair_count	경험치 복구량
money_get_count	돈 획득량
abyss	abyss 입장 횟수
use_portal_count	포탈 사용 횟수
killed_by PC/NPC	PC/NPC에게 죽은 횟수
teleport_count	텔레포트 사용횟수
reborn_count	부활 횟수
question_count	퀘스트 관련 횟수

1. Column 생성

Group activity

Column	사용된 로그 정보
total_party_time	총 파티 시간
guild_join_count	길드 활동 횟수
average_party_time	평균 파티 시간

1. Column 생성

Network_feature

Column	사용된 로그 정보
trade_go	거래 신청 보낸 횟수
trade_come	거래 신청 수락 횟수
buy_from_shop	상점에서 구입한 횟수
sell_to_shop	상점에 판매한 횟수
buy_from_personal_shop	개인상점에서 구입한 횟수
mail_go	우편을 보낸 횟수
mail_come	우편을 받은 횟수
invite	파티 초대 신청을 보낸 횟수
join	파티 초대를 수락한 횟수
revival	부활 횟수
fight	결투 횟수
friend_num	친구 수
friend_num_change_count	친구 수 변화량
too_much_gain	시간당 획득량이 과도한 경우
too_much_AP	같은 유저에게 AP를 과도하게 받은 경우

1. Column 생성

Per_day

Play_time < Play_time_per_day

(0.143) (0.375)

➡ 하루 평균 플레이 시간

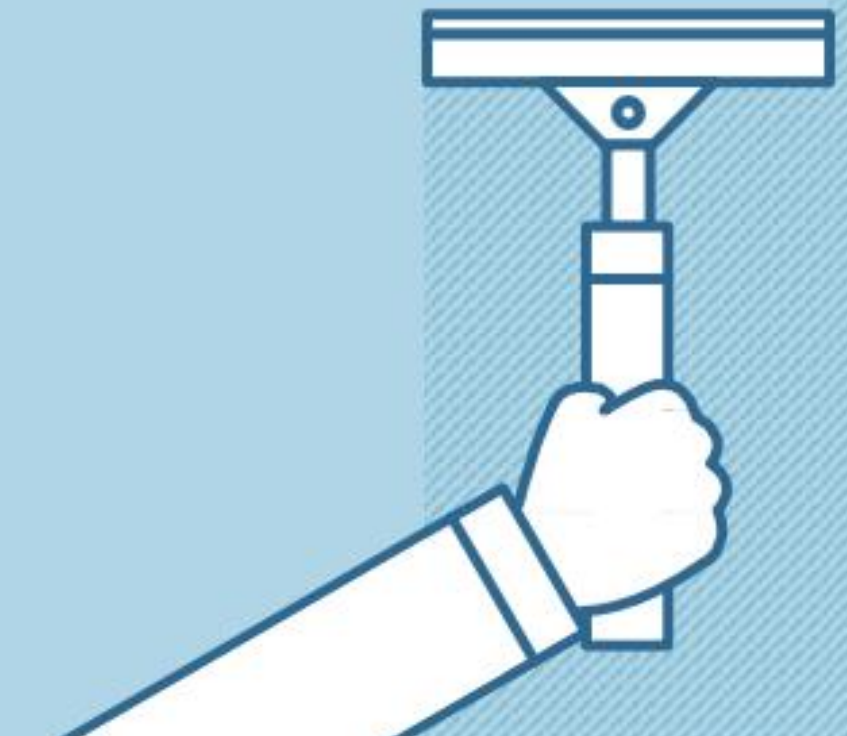
Sit_count < Sit_count_per_day

(0.124) (0.245)

➡ 하루 평균 앉은 횟수

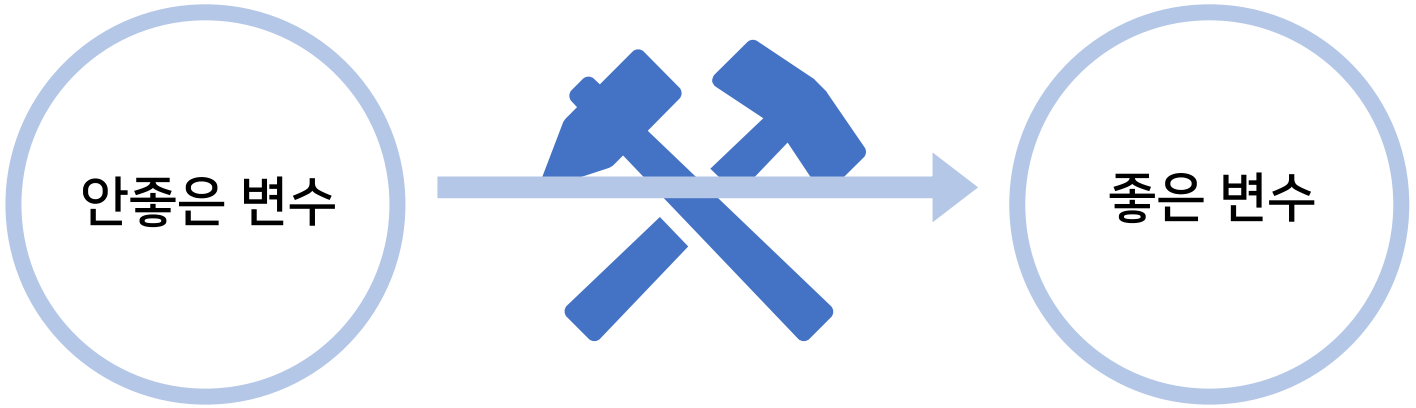
Column 재구성

피어슨 상관계수와 Lasso을 활용한 재구성



2. Column 재구성

가정



2. Column 재구성

Column 재구성

	login_count	logout_count	play_time	. . .	avg_party_t
login_count	login_count / login_count	logout_count / login_count	play_time / login_count		avg_party_t / login_count
logout_count	login_count / logout_count	logout_count / logout_count	play_time / logout_count		avg_party_t / logout_count
.					
.					
.					
avg_party_t	login_count / avg_party_t	logout_count / avg_party_t	Play_time / avg_party_t		avg_party_t / avg_party_t

2. Column 재구성

Column 재구성

	login_count	logout_count	play_time	. . .	avg_party_t
login_count	$\frac{\text{login\_count}}{\text{login\_count}}$	$\frac{\text{logout\_count}}{\text{login\_count}}$	$\frac{\text{play\_time}}{\text{login\_count}}$		$\frac{\text{avg\_party\_t}}{\text{login\_count}}$
logout_count	$\frac{\text{login\_count}}{\text{logout\_count}}$	$\frac{\text{logout\_count}}{\text{logout\_count}}$	$\frac{\text{play\_time}}{\text{logout\_count}}$		$\frac{\text{avg\_party\_t}}{\text{logout\_count}}$
.					
.					
.					
avg_party_t	$\frac{\text{login\_count}}{\text{avg\_party\_t}}$	$\frac{\text{logout\_count}}{\text{avg\_party\_t}}$	$\frac{\text{Play\_time}}{\text{avg\_party\_t}}$		$\frac{\text{avg\_party\_t}}{\text{avg\_party\_t}}$

58^2개의 변수 생성

2. Column 재구성

피어슨 상관계수

피어슨 상관관계 분석은 독립변수와 종속변수가 비례관계 가정하에 사용할 수 있다.
피어슨 상관계수 절대값이 1에 가까울 수록 상관관계가 큰 것이다.

$$r_{XY} = \frac{X\text{와 } Y\text{가 함께 변하는 정도}}{X\text{와 } Y\text{가 각각 변하는 정도}} = \frac{S_{XY}}{S_X S_Y}$$

➡ 피어슨 상관계수 절대값이 0.1 이상인 변수와 중복된 의미를 제거하여 최종 변수 54개 선택할 수 있었다.

2. Column 재구성

Lasso

Lasso는 중요한 몇 개의 변수만 선택하고 다른 계수들은 0으로 줄이는 feature selection을 한다.

$$\boldsymbol{min}(\|\boldsymbol{Y}-\boldsymbol{X\theta}\|_{\boldsymbol{2}}^{\boldsymbol{2}}+\lambda\|\boldsymbol{\theta}\|_{\boldsymbol{1}})$$

➡ Lasso을 활용하여 48개의 변수를 선택할 수 있었다.

2. Column 재구성

최종선택과정

Lasso로 산출한 48개의 변수와 Pearson 상관관계로 산출한 54개의 변수들을 서로 비교해 양측에서 모두 저조한 성적을 거둔 변수를 제거. 추가로 의미를 분석해 정보가 중복된다고 판단되는 정보를 제거. 51개의 최종 변수 선택.

Ex) money_get_count/play_time vs money_get_perday: 경우 단위 시간 당 얻는 돈의 규모에 대한 정보를 둘 다 포함하기에 하나는 제외한다.

2. Column 재구성

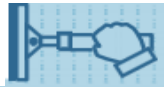
최종 사용 column

login_count
play_time
max_level
playtime_per_day
abyss sit_count
exp_get_amout
money_get_count
teleport_count
killed_by_pc

killed_by_npc
reborn_count
login_total_day
sit_count_perday
login_total_day
item_get_count_perday
money_get_count_perday
use_portal_count_perday
teleport_count_perday

total_party_time
sit_count/play_time
killed_by_pc/play_time
killed_by_npc/play_time
total-party_time/play_time
teleport_count/play_time
play_time/login_count
exp_get_amount/play_time
playtime_per_day/login_total_day

exp_get_amount/play_time
playtime_per_day/login_total_d
max_level/login_count
reborn_count/login_count
sit_count/max_level
play_time/question_count
… 총 51개



2. Column 재구성

기존 변수를 가공을 통한 효과

killed_by_pc (-0.034) < killed_by_pc/play_time (-0.088)

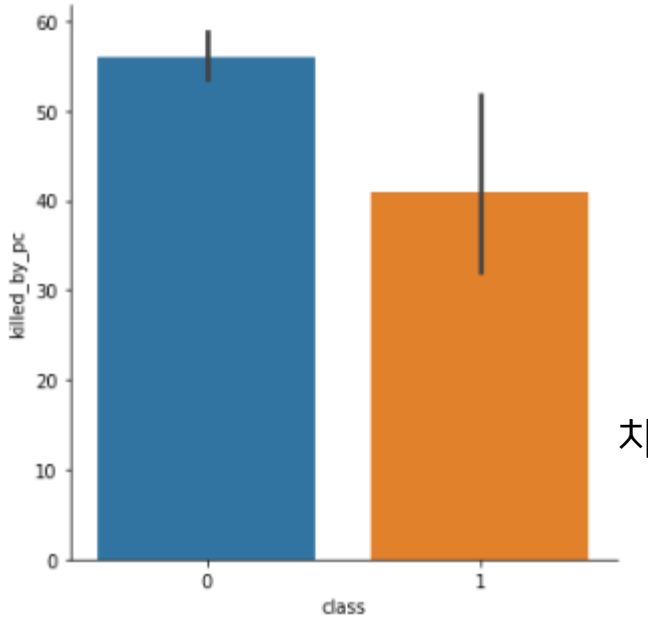
➡ 단위 시간당 유저에게 죽은 횟수

max_level (0.091) < max_level/login_count (0.121)

➡ 봇은 로그인을 한번 하고 오래 게임을 지속하니 값이 높다.

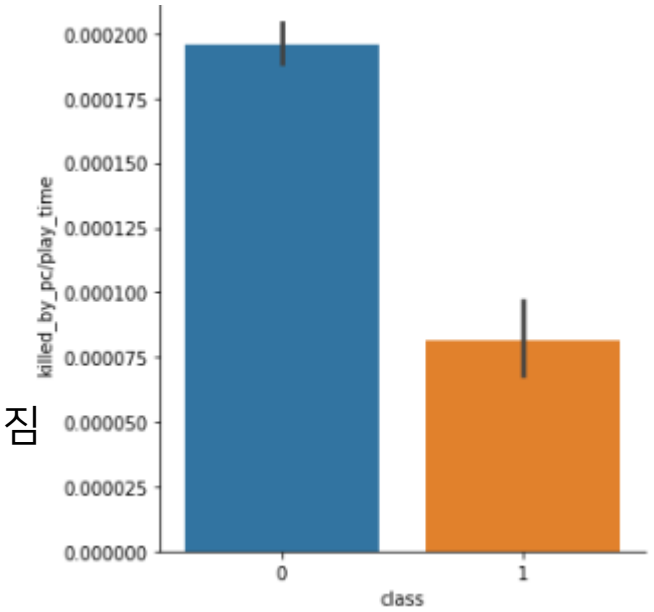
2. Column 재구성

기존 변수를 가공을 통한 효과



killed_by_pc

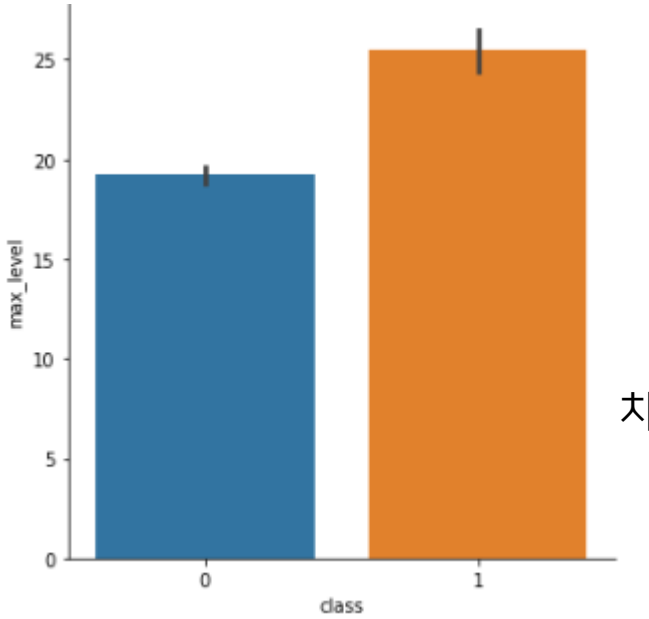
차이가 더 명확해짐



killed_by_pc/play_time

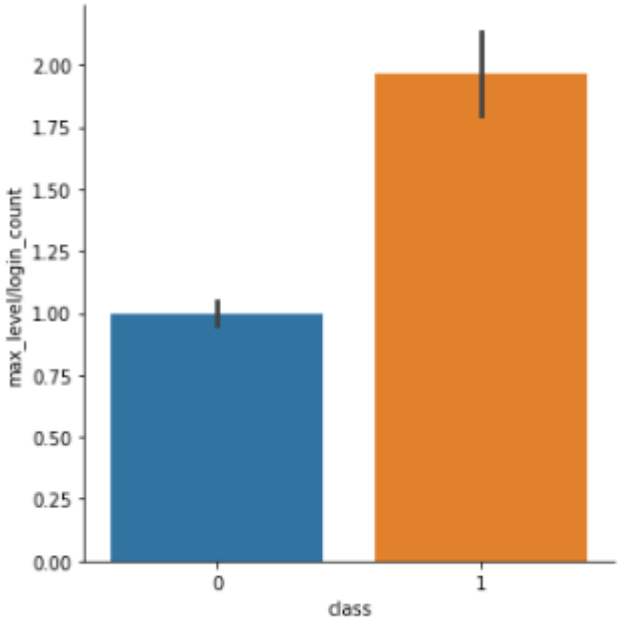
2. Column 재구성

기존 변수를 가공을 통한 효과



max_level

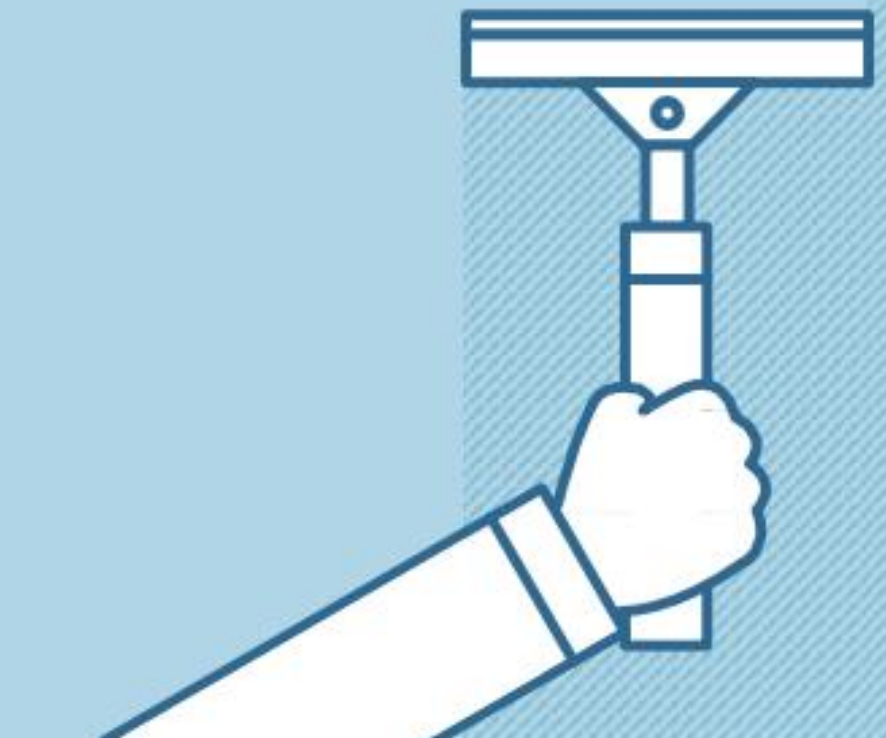
차이가 더 명확해짐



max_level/login_count

사용모델 및 성능평가

모델 사용 및 ROC-score



1. 데이터 구간

Raw data		Used data
예선데이터 2주	→	예선데이터 1주차
	→	예선데이터 2주차
본선데이터 1주	→	본선데이터 1주

2. 모델 적용

사용모델

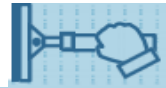
Xgboost, RandomForset,
Catboost

Xgboost_plot_importance

각 column들의 성능 기여도를 파악

Grid search

최적의 파라미터를 계산



3. 성능평가

GridSearchCV

```
[ ] xgb_model = xgb.XGBClassifier(learning_rate =0.001,
                                  n_jobs=-1,
                                  n_estimators=100,
                                  max_depth=15,
                                  min_child_weight=2,
                                  gamma=0,
                                  #n_estimators=2000,
                                  subsample=0.9,
                                  colsample_bytree=0.9,
                                  missing=-999,
                                  tree_method='gpu_hist')
```

```
[ ] gamma_test = {'gamma':[i/100.0 for i in range(0,6)]}

    gsearch2 = GridSearchCV(estimator = xgb_model, param_grid = gamma_test, scoring='roc_auc', cv=5)

    gsearch2.fit(X_train,y_train)

    gsearch2.cv_results_['params'], gsearch2.best_params_, gsearch2.best_score_
```

```
([{'gamma': 0.0},
  {'gamma': 0.01},
  {'gamma': 0.02},
  {'gamma': 0.03},
  {'gamma': 0.04},
  {'gamma': 0.05}],
 {'gamma': 0.02},
 0.8769353034565008)
```

Grid Search를 통해서
parameter의 최적값을 계산

```
rnd_clf = RandomForestClassifier(
  bootstrap = False,
  max_depth = None,
  max_features= 'sqrt',
  max_leaf_nodes = None,
  min_impurity_decrease = 0.0,
  min_impurity_split = None,
  min_samples_leaf = 1,
  min_samples_split = 2,
  min_weight_fraction_leaf = 0.0,
  n_estimators = 2000,
  n_jobs = 1,
  oob_score = False,
  random_state = 42,
  verbose = 0,
  warm_start = False)
```

3. 성능평가

Catboost, Random Forset

	Catboost	Random Forest
F1 score	75.043%	78.521%

감사합니다