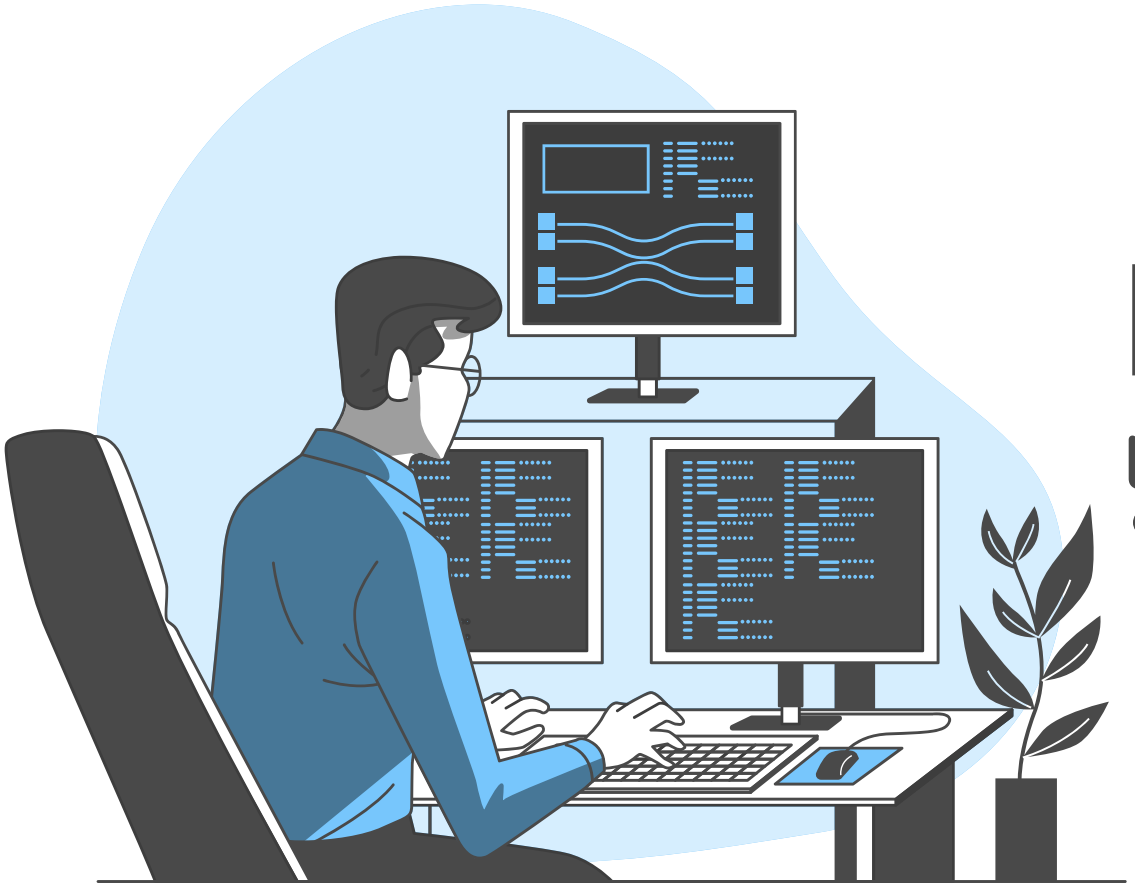


DCSS의 미래 방향성 제시

업데이트 주기에 대한 예측을 바탕으로

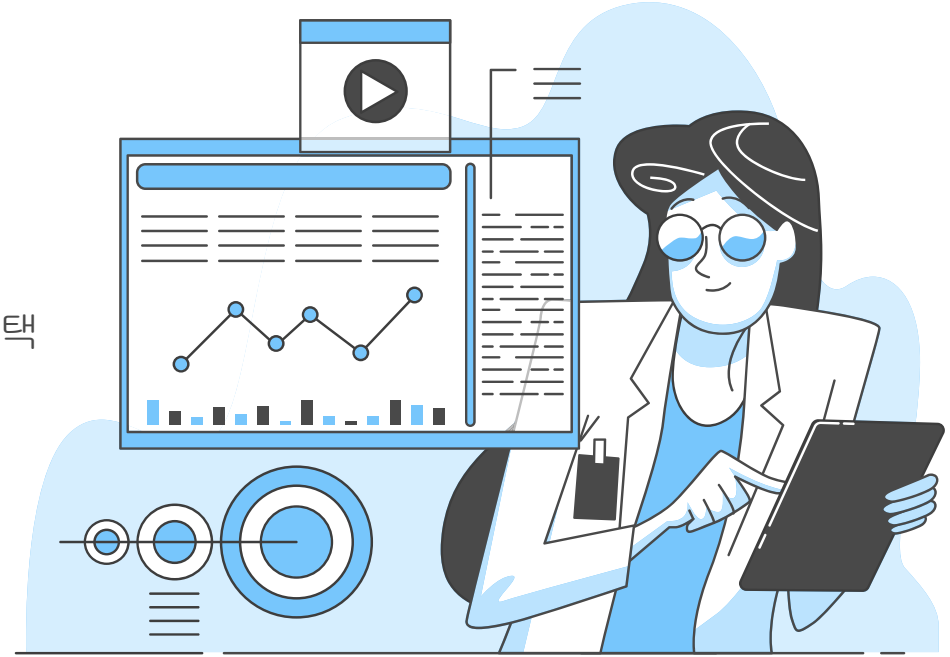
목차

01 상황 및 문제점
DCSS의 상황과 문제를 제시한다.

02 해결방안 제시
한계효용체감 법칙을 토대로

03 모델학습
회귀 문제에 대한 모델 선택

04 결론 제안
모델 해석으로 합리적인 결론 도출



01

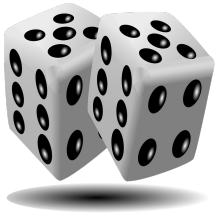
상황 및 문제점

DCSS의 유저가 떠나간다!

상황

- DCSS는 잘 알려지지 않은 게임이지만, 상당히 매니악한 유저층을 가지고 있다.
- DCSS는 로그라이크 장르고, 이 장르는 확실하게 정의되진 않았지만 공통적으로 "절차적 생성" 을 핵심으로 가지고 있다.

로그라이크의 이해



랜덤요소

절차적 생성으로
맵을 만들고 각종
요소들을
랜덤하게 만든다.
…

영구적 죽음

캐릭터가 사망할
경우 다시
살아나지 않는다.
…

성장억제

여러 번 죽음을
반복해도 그것이
이득을 주지
않는다.
…

이것들은 모두 유저들을 **“덜 지루하게”** 만들기 위한 방법입니다

지루함은 필연적으로 찾아온다

01

숙련도

경험이 반복되면
결국 지루해진다

02

절차적 생성의 한계

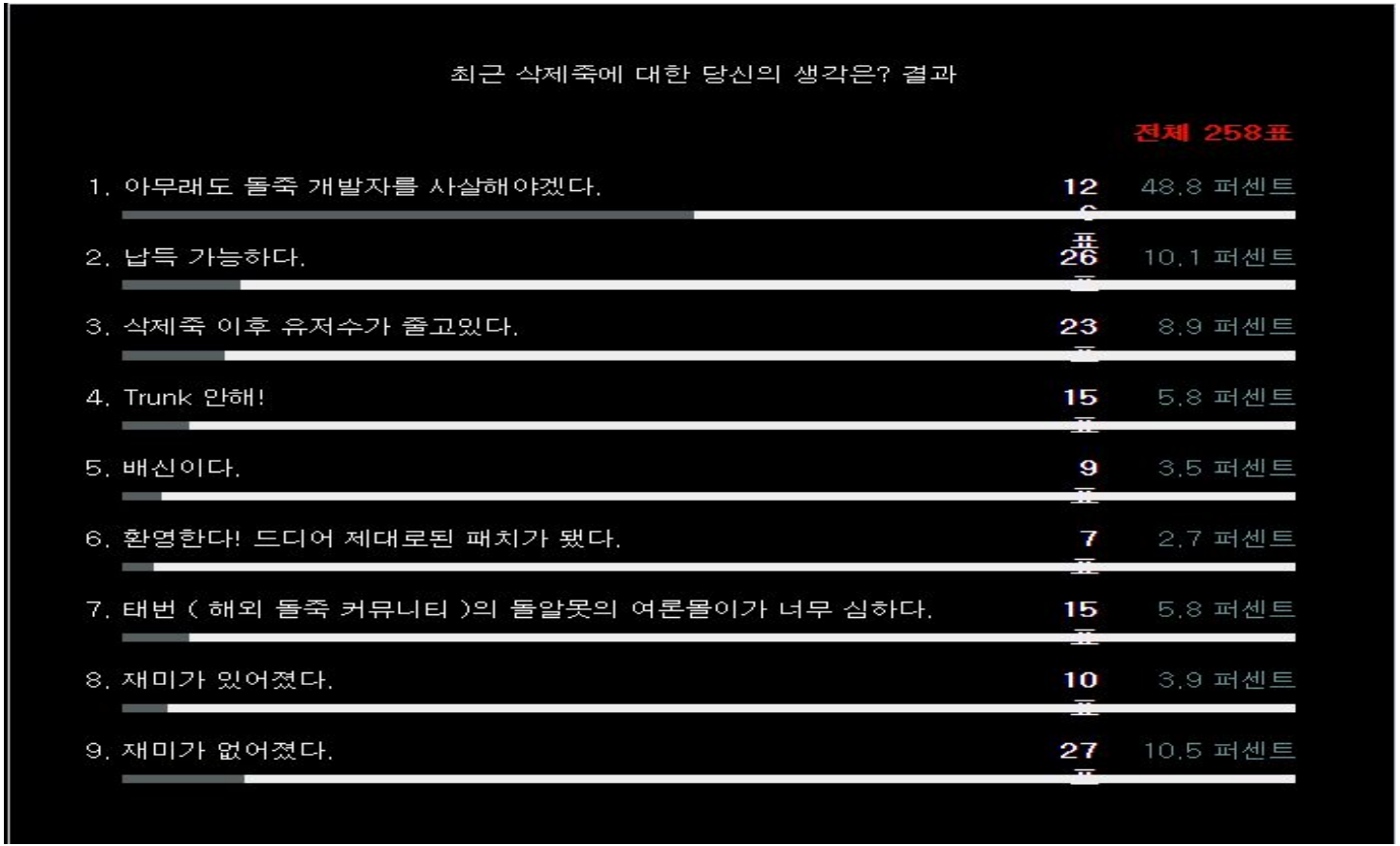
아무리 랜덤하게
만들었어도
반복되는 난수의
패턴은 보이게 된다

03

일반적이지 않은 유저층

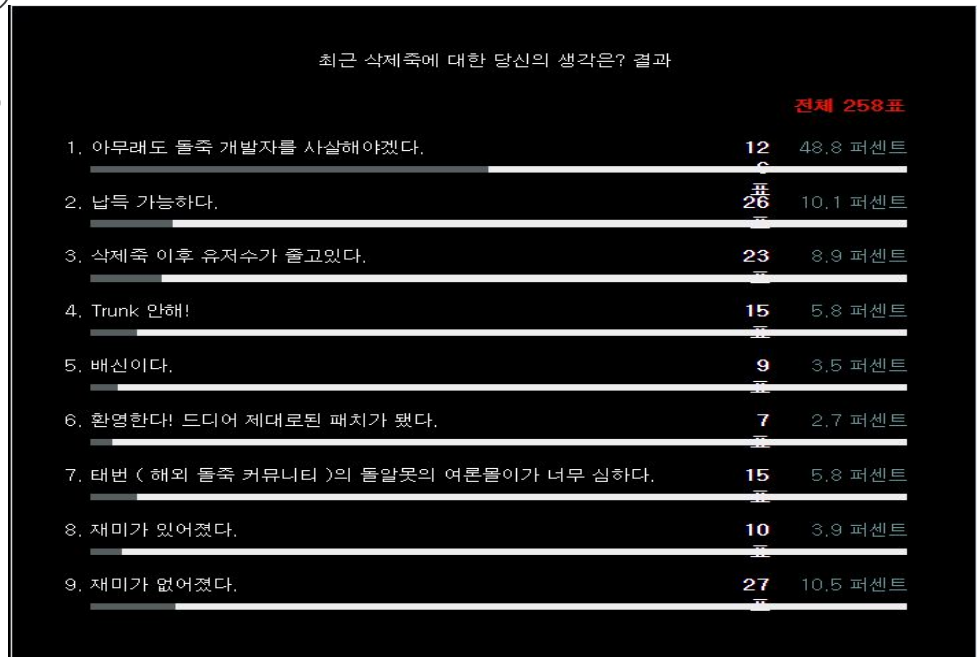
이런 장르를 즐기는
사람들은 게임이
주는 좌절감과
패배감을 즐긴다.

현 DCSS의 문제점


최근 삭제죽에 대한 당신의 생각은? 결과
전체 258표
1. 아무래도 돌죽 개발자를 사살해야겠다. 126표 48.8 퍼센트
2. 납득 가능하다. 26표 10.1 퍼센트
3. 삭제죽 이후 유저수가 줄고있다. 23표 8.9 퍼센트
4. Trunk 안해! 15표 5.8 퍼센트
5. 배신이다. 9표 3.5 퍼센트
6. 환영한다! 드디어 제대로된 패치가 됐다. 7표 2.7 퍼센트
7. 태번 (해외 돌죽 커뮤니티)의 돌알못의 여론몰이가 너무 심하다. 15표 5.8 퍼센트
8. 재미가 있어졌다. 10표 3.9 퍼센트
9. 재미가 없어졌다. 27표 10.5 퍼센트

업데이트에 불만이 많다

설문조사



77.5%

명백한 반대 의사를 표출

6.6%

명백한 찬성 의견

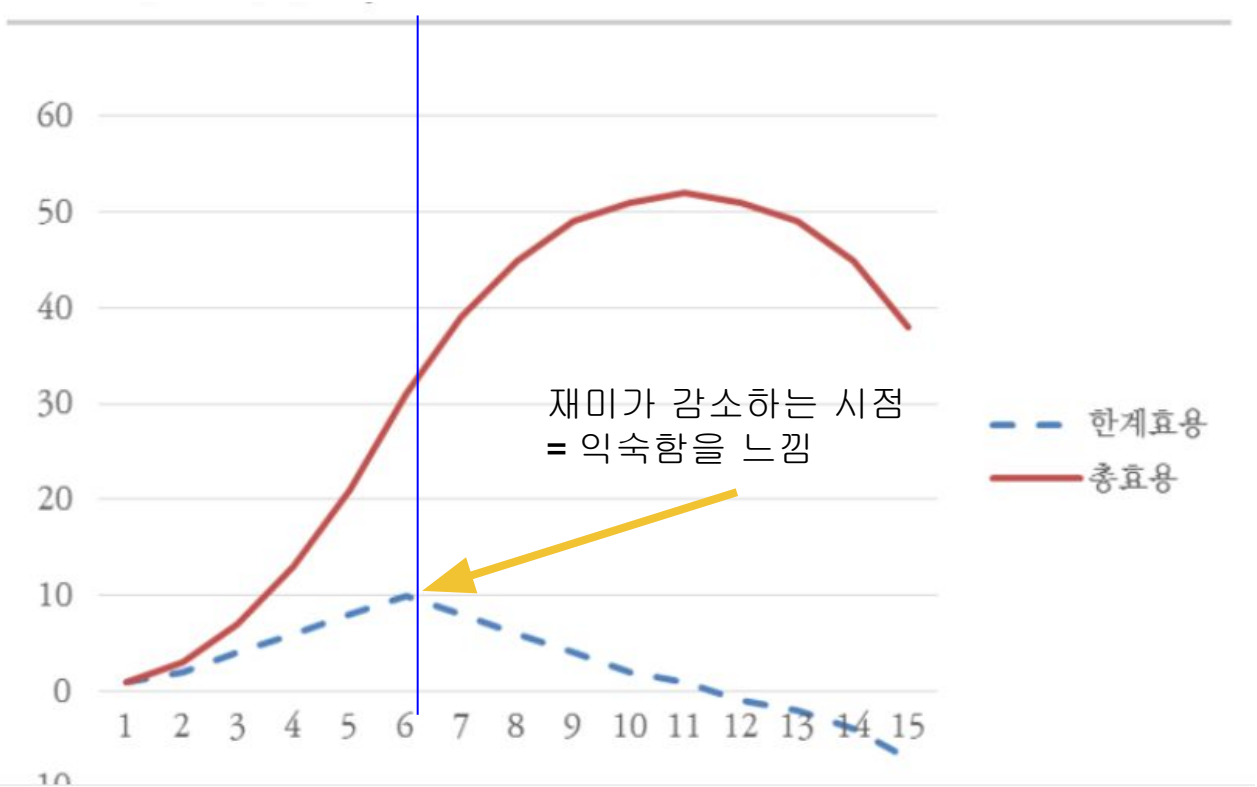
"가장 불행한 고객은
가장 큰 배움의 원천입니다."

–빌 게이츠

02

해결방안 제시

한계효용체감의 법칙


60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
재미가 감소하는 시점
= 익숙함을 느낌
한계효용
총효용

DCSS 공식 데이터

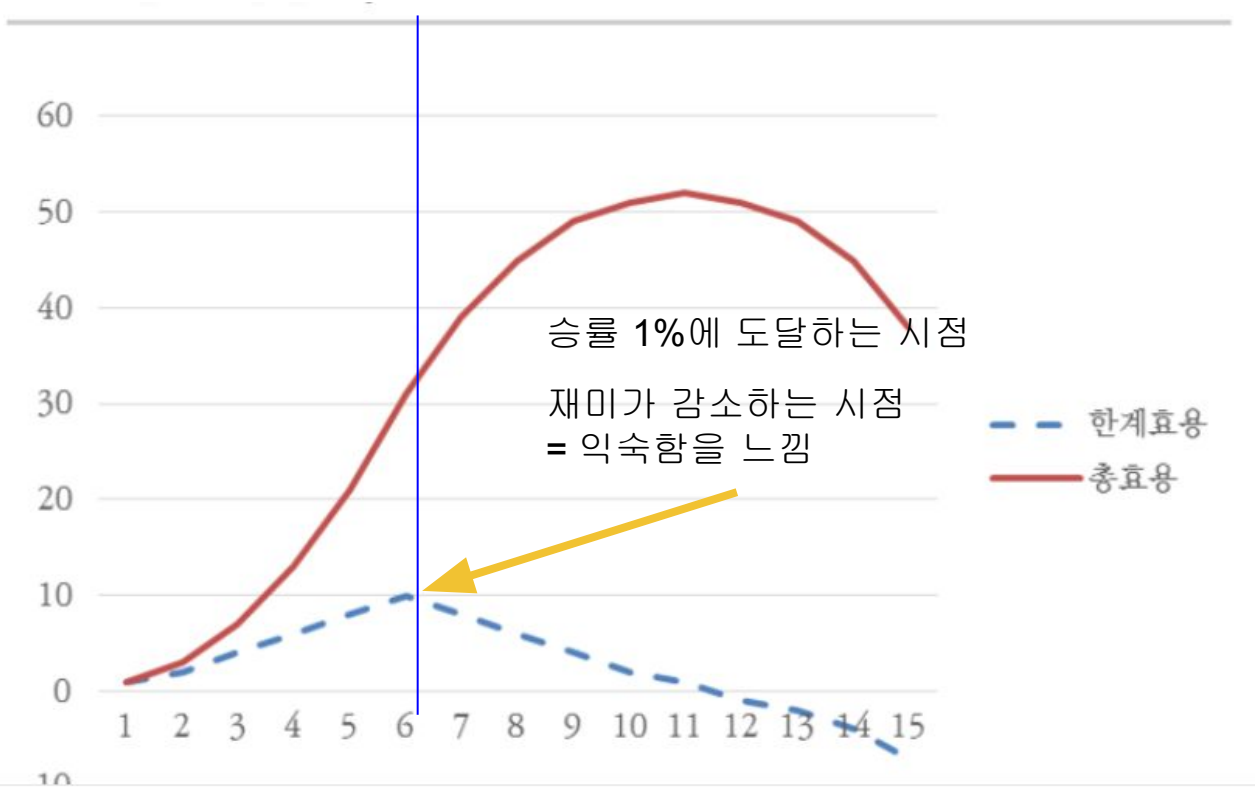
숙련도 = 승률

2022-09	134213	4444	1682
2022-10	122459	4112	1298
2022-11	110222	3877	1104

일반적인 유저들의 승률은 대략 1%

따라서 **승률 1%를 최소한의 숙련도 기준**으로 잡고
승률 1%에 도달하는 시점에서 특성을 알아보도록 한다.
이는 **회귀 문제**가 될 것이고, MAE, RMSE, R2의 평가지표를 사용한다.

한계효용체감의 법칙


60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
승률 1%에 도달하는 시점
재미가 감소하는 시점
= 익숙함을 느낌
한계효용
총효용

DCSS 공식 데이터

모든 유저에 대한 데이터를 분석

All Players

This page may not show accounts that have been inactive for more than a year.

	Total Score	Player	Games Played	Games Won	Win %	Best XL	Best Score	Average Score	First Game	Most Recent Game
1	133,003	00000	4	0	0.00%	16	129,784	33,250	2013-06-17 10:55:38	2013-06-18 07:48:31
2	1,352,041	00000ooooo	5	0	0.00%	27	722,654	270,408	2018-01-01 12:47:54	2018-01-04 10:33:21
3	256,252	0001mH	17	0	0.00%	16	111,895	15,073	2021-09-23 09:47:42	2022-05-11 22:30:36
4	782,760	000ooo000	4	0	0.00%	20	284,346	195,690	2017-12-19 11:53:36	2017-12-21 07:06:23
5	1,042,427	0098kg	49	0	0.00%	27	697,811	21,274	2019-01-03 23:20:39	2020-09-11 01:47:25
6	4,547,028	0099kg	235	0	0.00%	27	873,943	19,349	2019-01-21 09:10:56	2020-06-26 11:47:59
7	314,363	00gogo00	120	0	0.00%	18	174,449	2,619	2018-12-17 01:11:02	2022-07-25 21:56:12
8	38,099,219	00kami	28	7	25.00%	27	12,170,655	1,360,686	2020-10-08 11:17:00	2021-01-19 21:49:02
9	62,676,987	00Nightcrawler	628	5	0.80%	27	16,965,711	99,804	2016-11-04 17:31:25	2019-07-02 12:47:00
10	45,084,017	00Nightcrawler00	482	3	0.62%	27	17,959,522	93,535	2016-09-02 03:15:10	2017-02-09 21:15:04
11	2,401,185	0101son	195	0	0.00%	23	392,583	12,313	2016-10-10 01:35:10	2022-07-27 03:32:57
12	20,870,802	010203	151	9	5.96%	27	2,059,493	138,217	2019-06-13 06:30:46	2019-07-15 02:17:12

EDA 과정

```
target = 'Playtime'
df = df.eval('Odds = Games_won / Games_Played')
df['Total_score'] = df['Total_score'].replace(',','',regex=True).astype(int)
df['Best_Score'] = df['Best_Score'].replace(',','',regex=True).astype(int)
df = df.eval('Avg_Score = Total_score / Games_Played')
df['Avg_Score'] = round(df['Avg_Score'], 0).astype(int)
df['First_game'] = pd.to_datetime(df['First_game'])
df['Last_game'] = pd.to_datetime(df['Last_game'])
a = (df['Last_game'] - df['First_game']).dt.days * 86400
b = (df['Last_game'] - df['First_game']).dt.seconds
df['Playtime'] = a+b
df = df[df['Playtime'] > 0]
df = df[df['Odds'] < 1.0]
drop_cols = ['Win_per', 'Games_won', 'Average_Score','Player_name', 'First_game', 'Last_game', 'Best_XL']
df.drop(drop_cols, axis=1, inplace=True)
```

ames_won / Games_Played = Odds 라는 상관관계가 있기 때문에 Games_won은 drop해줍니다.

존의 Win_per도 삭제. 필요없는 특성은 다 지워줍니다.

연히 최대 레벨에 해당하는 Best_XL 또한 의미있는 지표가 아니며, Avg_Playtime이라는 평균 게임 시간을 새로 만들어줍니다.

aytime이 0이거나 0보다 작을 경우 데이터 수집에서 오류라고 판단합니다.

률이 100%인 경우, 오류거나 특출난 실력을 가졌다는 것이고 이는 예측하려는 방향성에서 벗어났기 때문에 삭제합니다.

특성공학 이후 데이터는

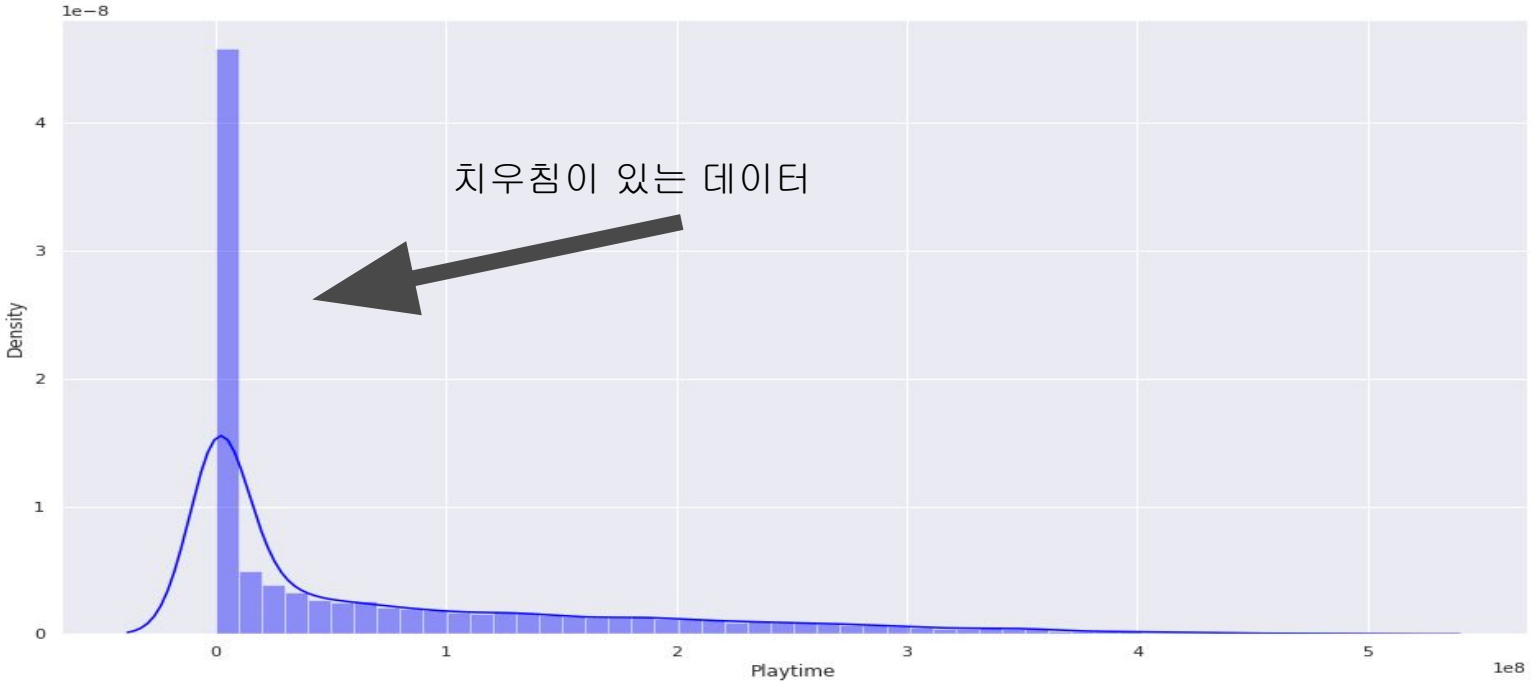
53865 명의 유저 데이터

6 개의 특성

이를 Train/Validation/Test 의 3개의 데이터셋으로 나눠줍니다.

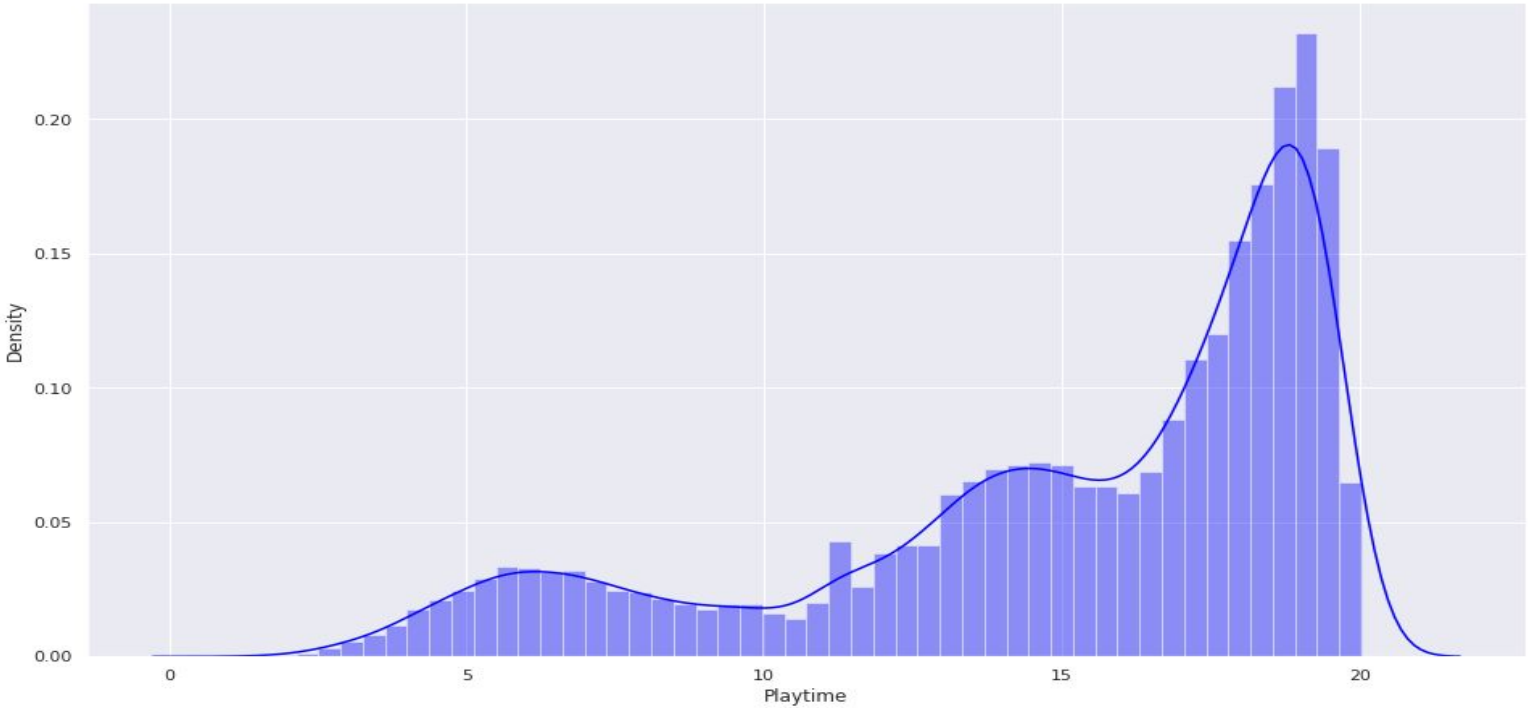
6:2:2의 비율로 나눴습니다.

Target(게임플레이 시간-Playtime)



타겟 데이터의 치우침 문제는 성능에 영향을 줄 가능성이 높습니다.
Right skewd 문제를 위해 로그 변환을 해주도록 하겠습니다.

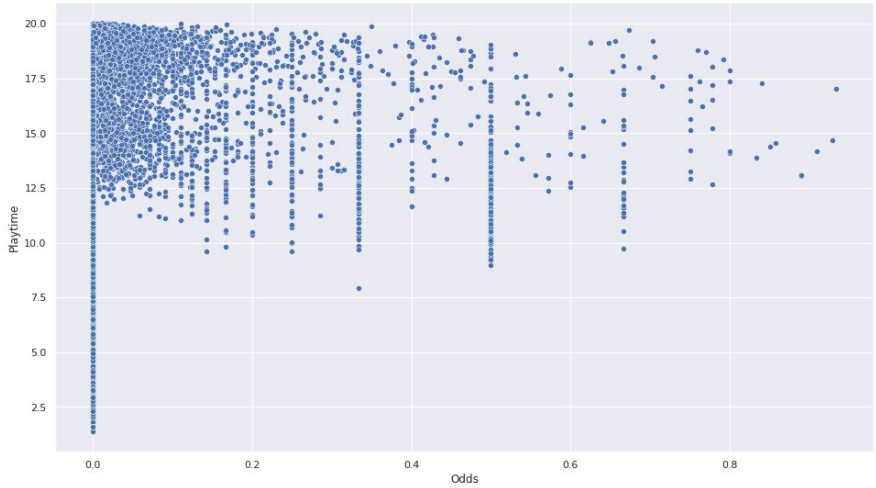
Target(게임플레이 시간-Playtime)



로그 변환 이후

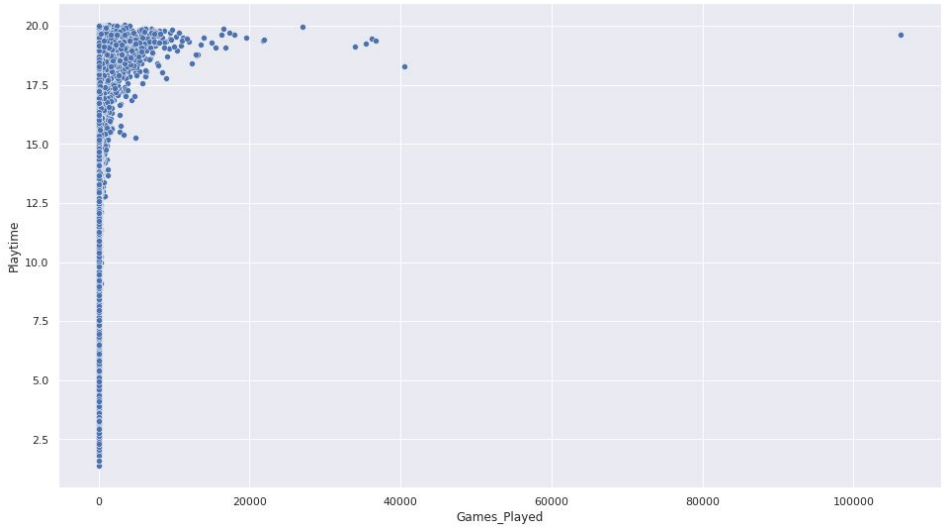
Target(Playtime)

승률 과의 관계



승률이 높을 때 Playtime이 높은 경우가 많다.

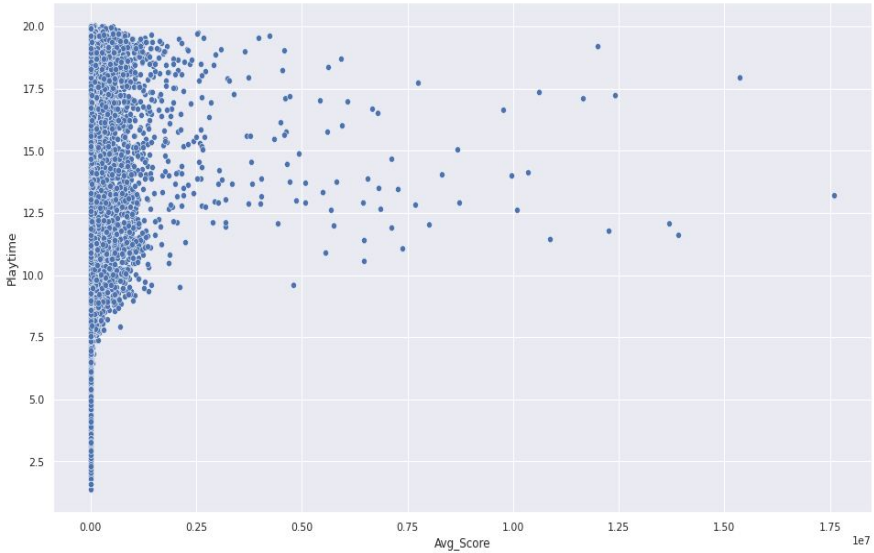
게임 판수와의 관계



게임 판수가 많으면 당연히 **Playtime**이 높다.
일반적인 생각과 비슷한 양상을 보인다.

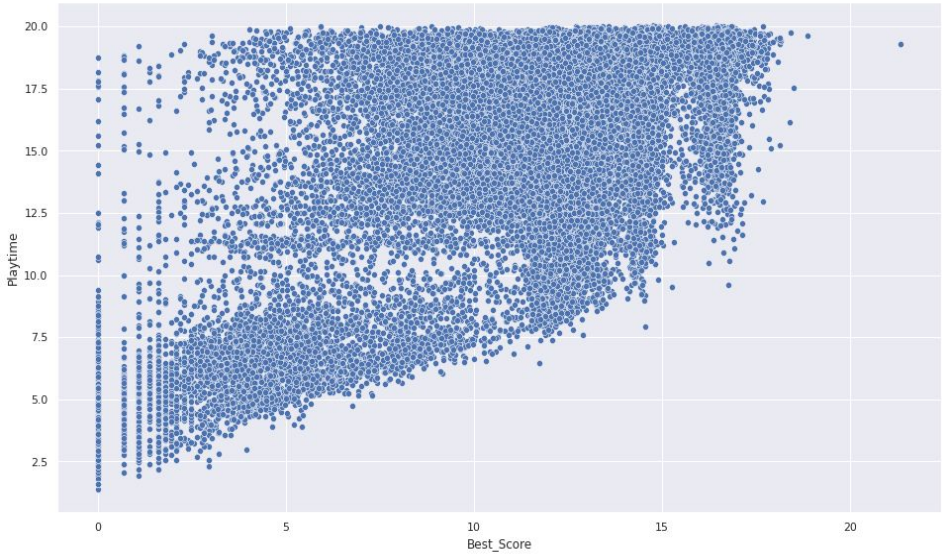
Target(Playtime)

평균 점수와의 관계



명확한 상관관계를 읽기는 힘드나,
평균 점수가 높은 경우 Playtime이 낮은 경우가 없다.

최고 점수와의 관계

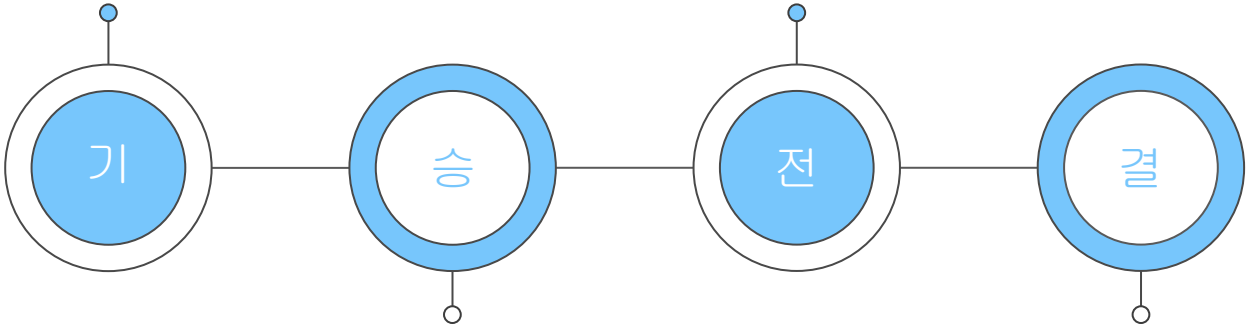


최고 점수가 높을수록 **Playtime**이 길어진다는 관계가 보인다.

머신러닝 모델 Workflow

기준 모델에서
평가지표 확인

테스트 데이터셋의
평가지표 확인



모델 선정 및 학습

PDP 및 Shap를 통해서
해석 후 결론 도출

03

모델 학습

평가지표

RMSE

Mean Square Error
이상치에 민감한
오차에 대한 지표

MAE

Mean Absolute Error
이상치에 둔감한
오차에 대한 지표

R2

독립변수가 종속변수를
잘 설명하는 지에 대한
지표

기준 모델

단순히 **예측한 승률이 모두 평균이라고 가정했을 때를 기준 모델**으로 선정

기준 모델보다는 좋은 지표가 나와야 모델을 사용하는 의미가 있습니다.

훈련 데이터

MAE : 3.59

RMSE : 4.411845026999232

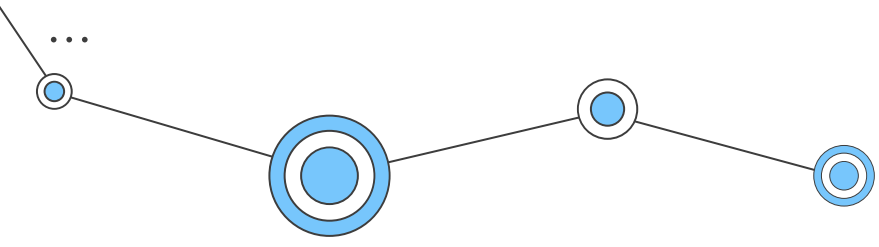
R2 : 0.0

평가 데이터

MAE : 3.64

RMSE : 4.479054183289326

R2 : -0.0004778274983880948



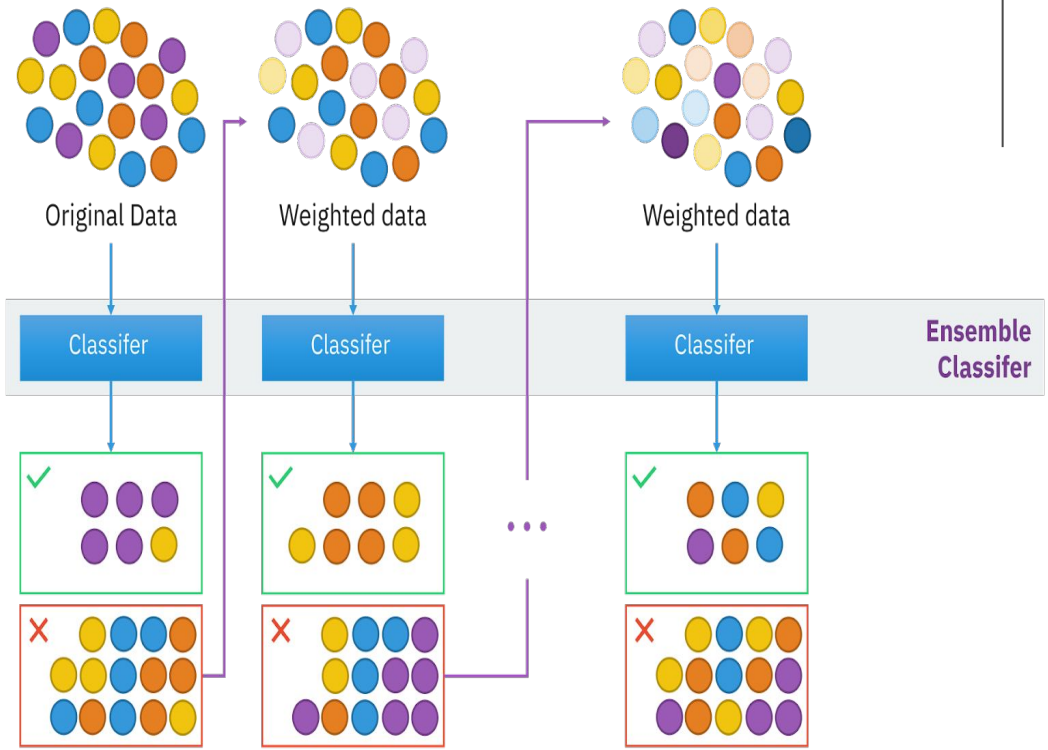
사용 모델

XGBoost

부스팅 기법으로 큰 데이터셋에서 좋은 성능을 보여줍니다

HyperOpt

모델의 세부 조절을 도와줍니다.



하이퍼 파라미터 조정

HyperOpt
모델의 세부 조절을 도와줍니다.

Cross Validation
K-fold 방식으로 평가지표의 정확도를 올립니다.

```
100%|██████████| 50/50 [34:21<00:00,
41.24s/it, best loss: 1.667022974216965]
{'colsample_bytree': 0.8085533354561016,
 'learning_rate': 0.031380826334965 67,
 'max_depth': 2.0,
 'min_child_weight': 4.0,
 'n_estimators': 810.0,
 'scale_pos_weight': 12.0}
```

colsample_bytree : 트리 생성에 필요한 특성 샘플링 비율을 정합니다
learning_rate : 모델의 학습 속도를 제어합니다.
max_depth : 트리구조의 최대 깊이를 결정합니다. 높을수록 더 복잡한 구조가 됩니다
Min_child_weight : 학습을 위한 가중치의 최소입니다. 높을수록 데이터를 덜 학습합니다.
n_estimators : 나무의 개수.
scale_pos_weight : 데이터가 불균형할때 사용합니다.

학습 모델 평가지표

기준 모델
MAE : 3.64
RMSE : 4.479054183289326
R2 : -0.0004778274983880948

학습 모델
MAE : 1.67
RMSE: 2.189283320395807
R2: 0.7609775065628036

모든 지표가 좋은 방향으로 나아진 것을 확인했습니다.
MAE, RMSE는 작을수록 좋은 성능을
R2는 높을수록 좋은 성능을 나타냅니다.

Test 데이터
MAE : 1.66
RMSE : 2.2

0.75530942

Test 데이터의 R2 score

R2는 1에 가까울수록
설명력이 좋다고
판단합니다.

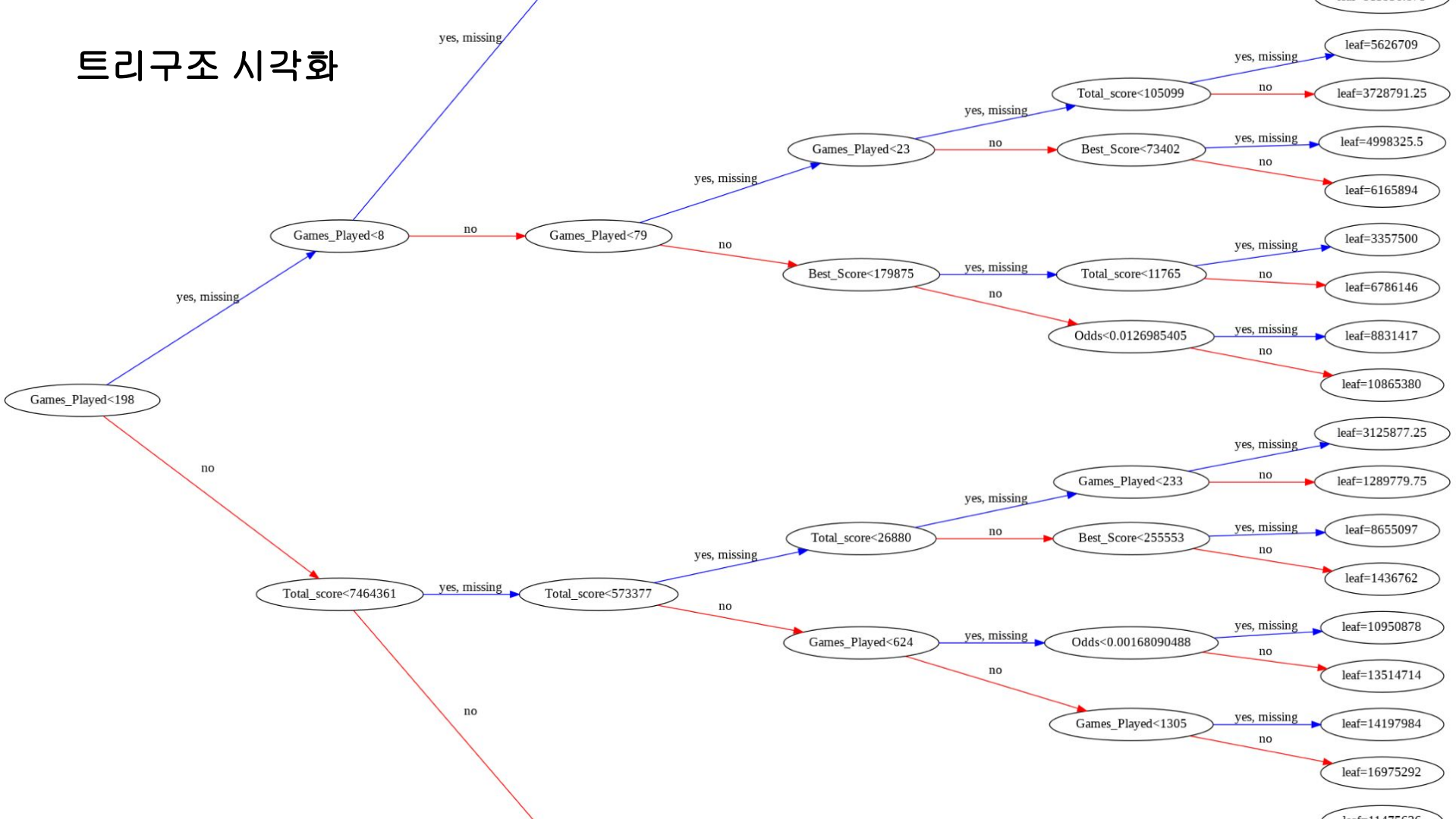
모델 해석

- 트리구조 시각화
- PDP

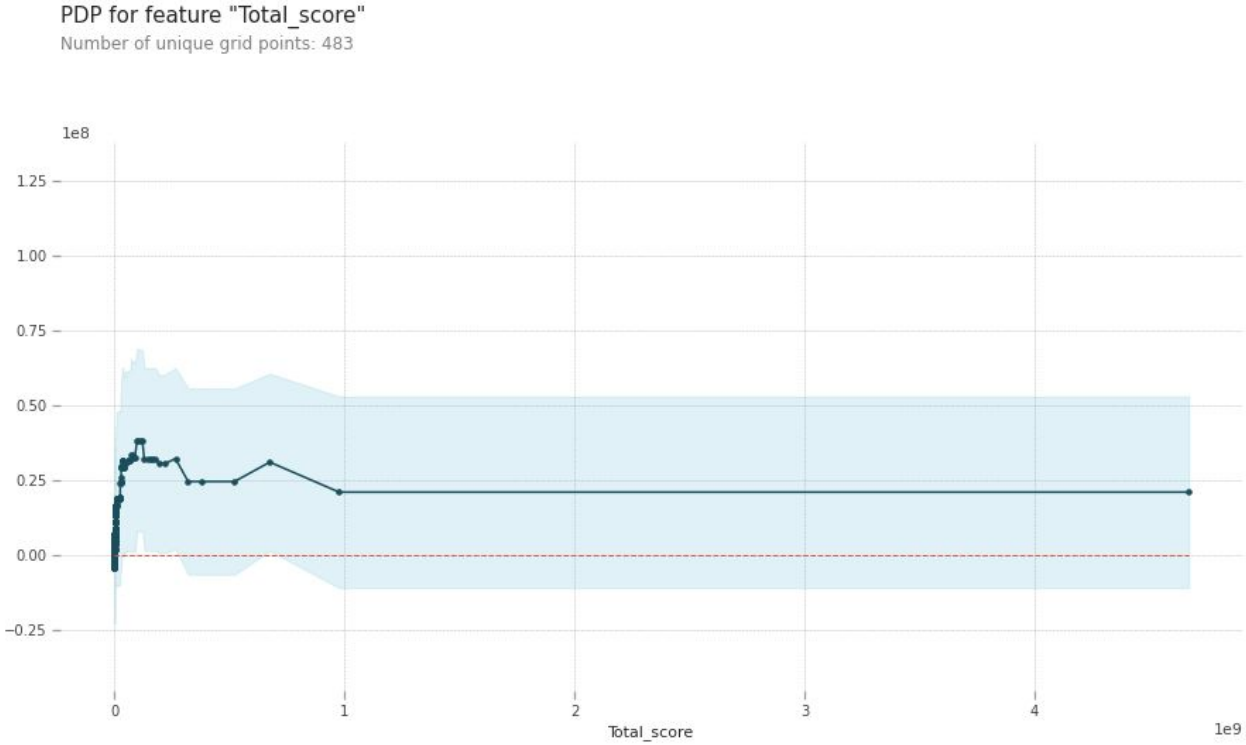
  특성별 모델 해석
- SHAP

  전체적인 관점에서의 모델 해석

트리구조 시각화

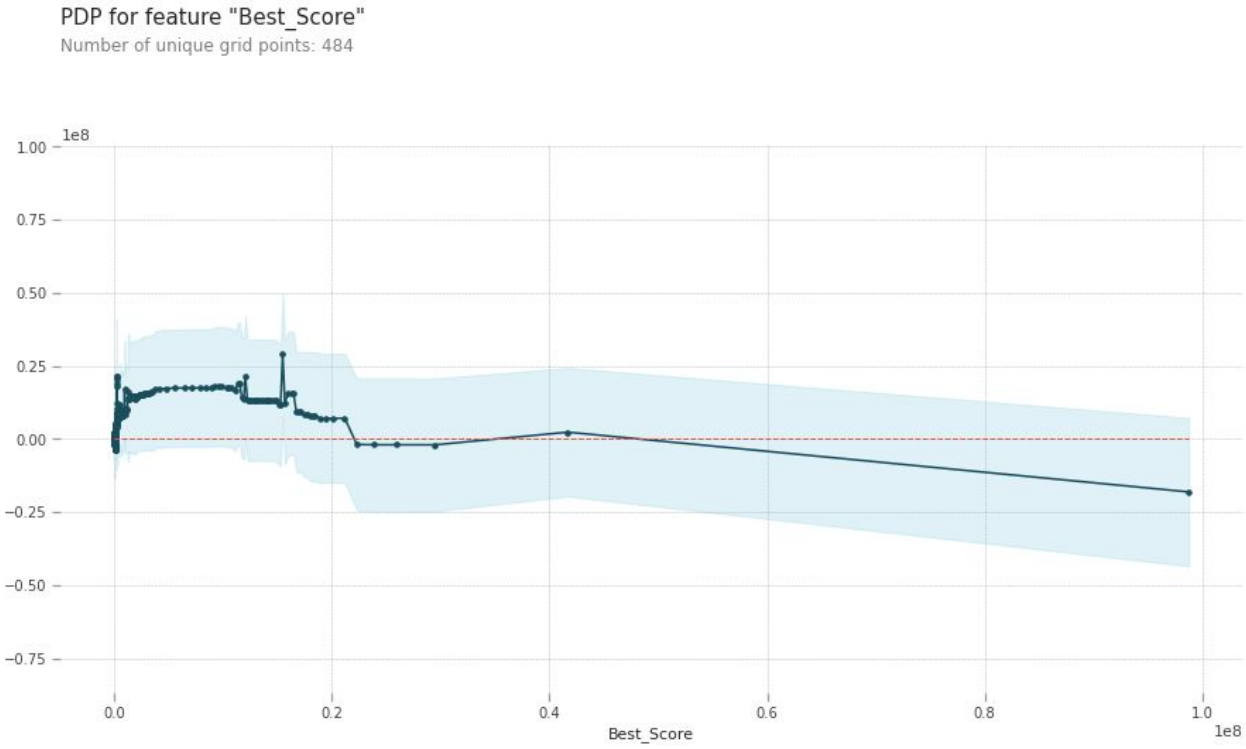

Games_Played<198
yes, missing
Games_Played<8
yes, missing
no
Games_Played<79
yes, missing
Games_Played<23
yes, missing
Total_score<105099
yes, missing
leaf=5626709
no
leaf=3728791.25
no
Best_Score<73402
yes, missing
leaf=4998325.5
no
leaf=6165894
no
Best_Score<179875
yes, missing
Total_score<11765
yes, missing
leaf=3357500
no
leaf=6786146
no
Odds<0.0126985405
yes, missing
leaf=8831417
no
leaf=10865380
no
Total_score<7464361
yes, missing
Total_score<573377
yes, missing
Total_score<26880
yes, missing
Games_Played<233
yes, missing
leaf=3125877.25
no
leaf=1289779.75
no
Best_Score<255553
yes, missing
leaf=8655097
no
leaf=1436762
no
Games_Played<624
yes, missing
Odds<0.00168090488
yes, missing
leaf=10950878
no
leaf=13514714
no
Games_Played<1305
yes, missing
leaf=14197984
no
leaf=16975292
no

PDP



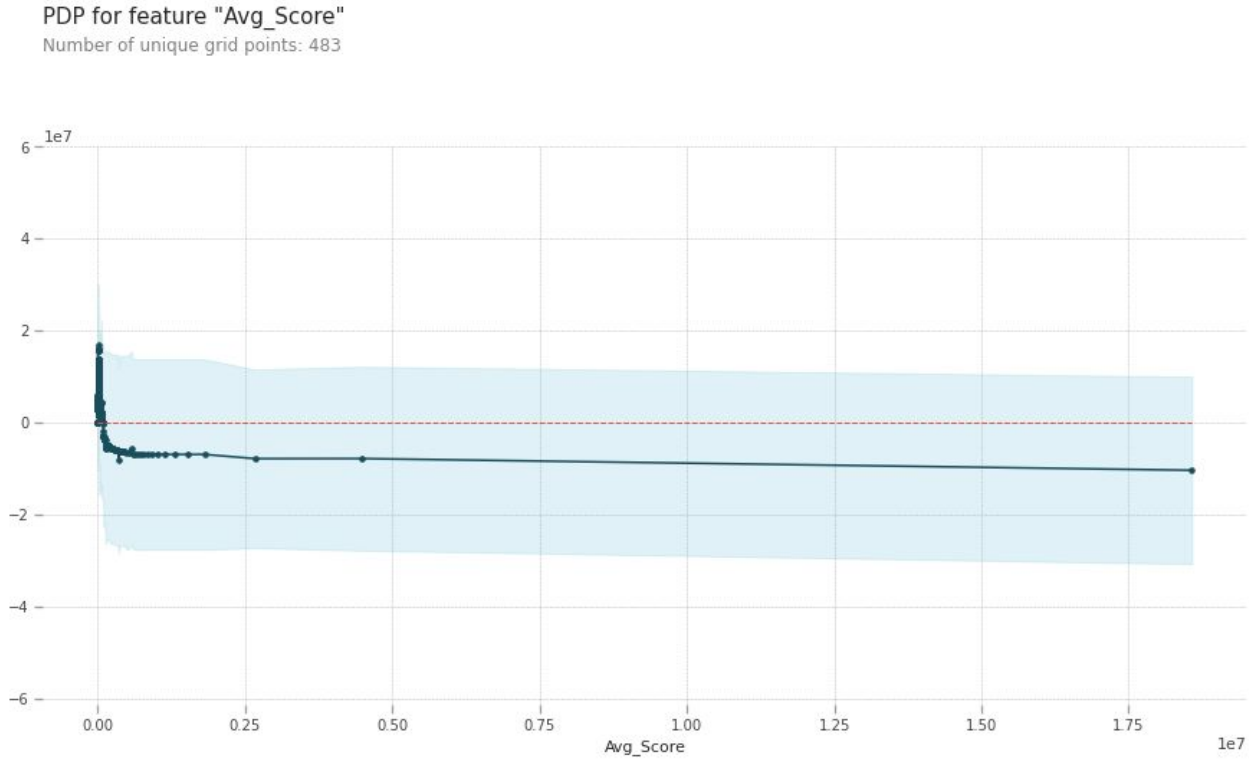
총합 점수 (Total_score)는 당연하게도 높을수록 타겟에 영향을 많이 주고 있다. 하지만, 상승 그래프는 아니다.

PDP



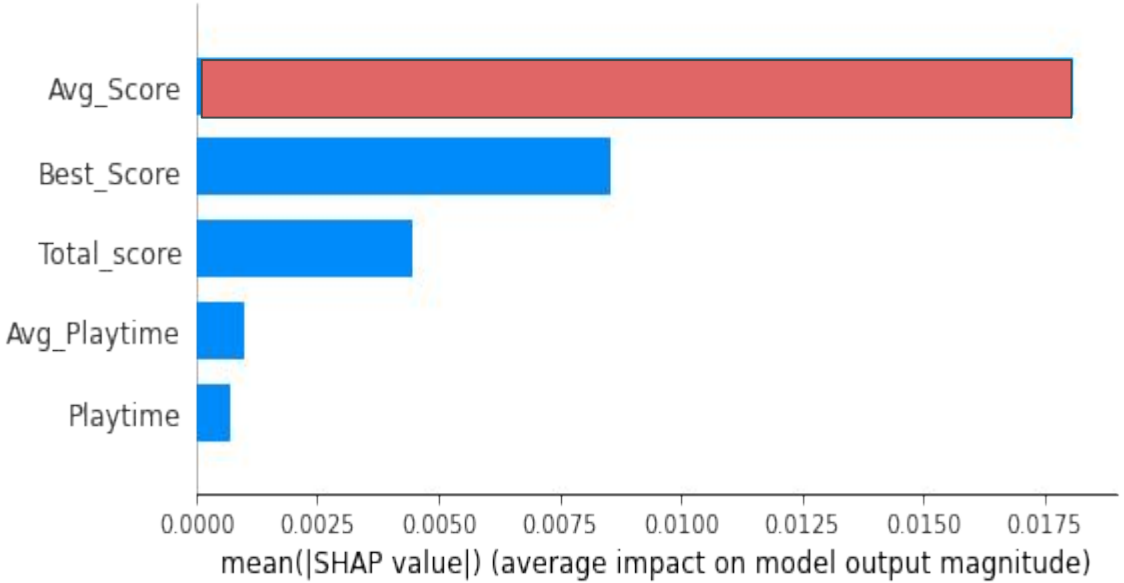
Best_score(최대 점수)는 일정 구간에서는 타겟에게 영향을 주지만, 이후로는 전혀 상관이 없다.

PDP



Avg_score(평균 점수)는 높을수록 타겟을 감소시키는 영향을 준다. 어째서일까?

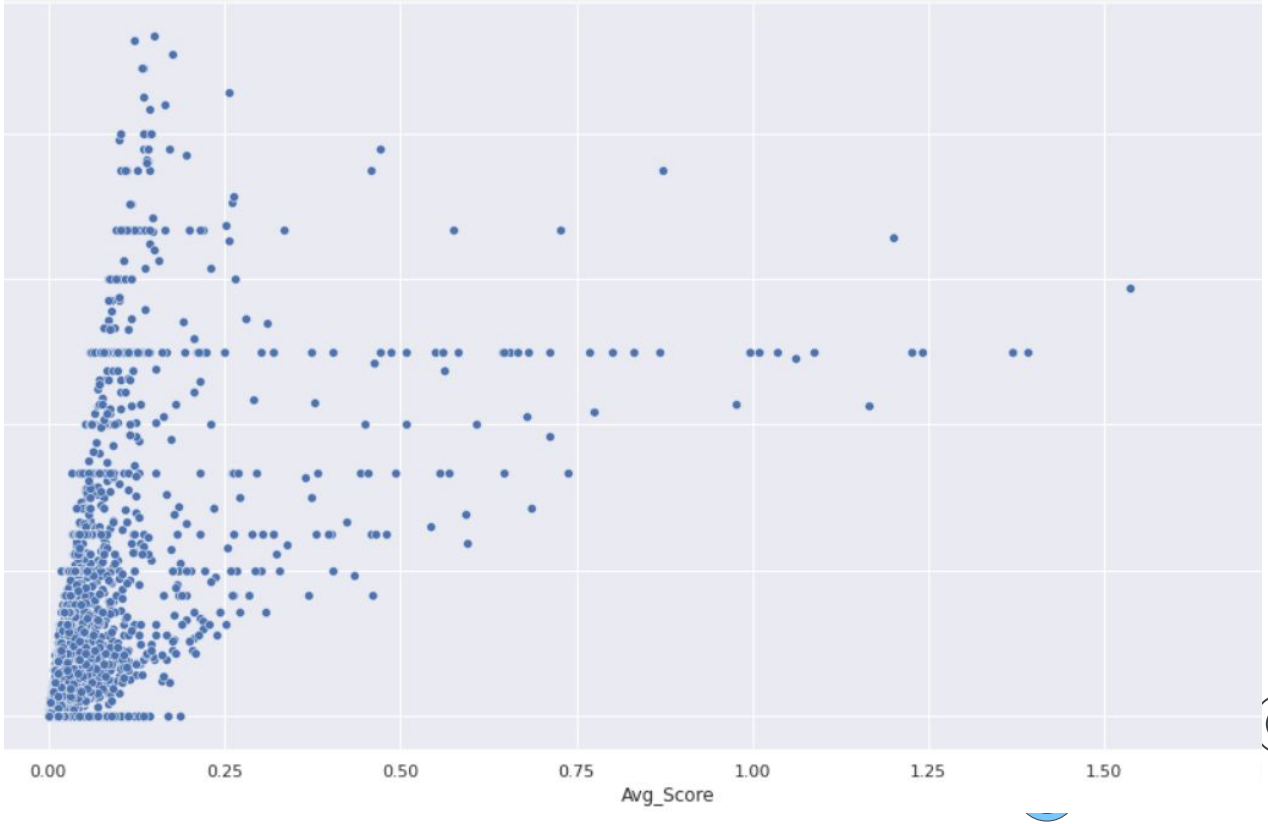
SHAP 특성 중요도
Target=”승률”



Avg_Score

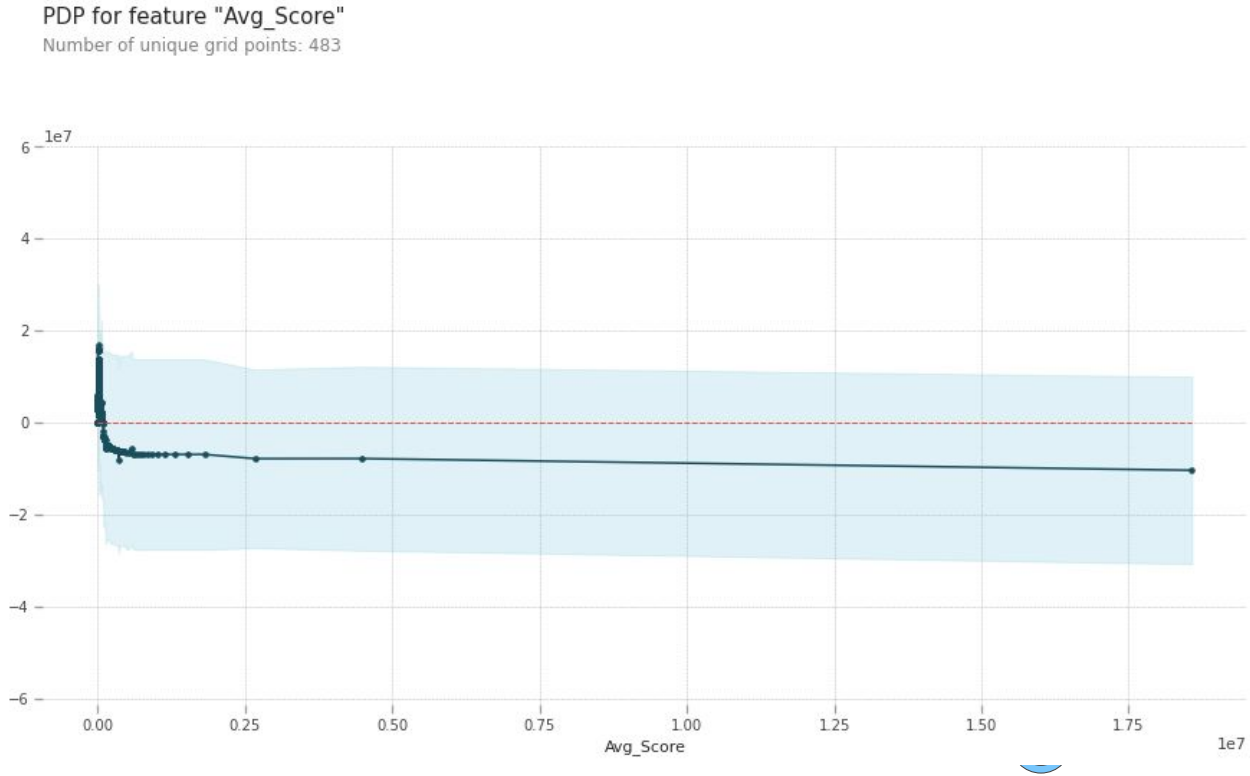
압도적으로 중요도가 높은 특성이다. 승률과 아주 강한 관계가 있다고 볼 수 있다.

PDP



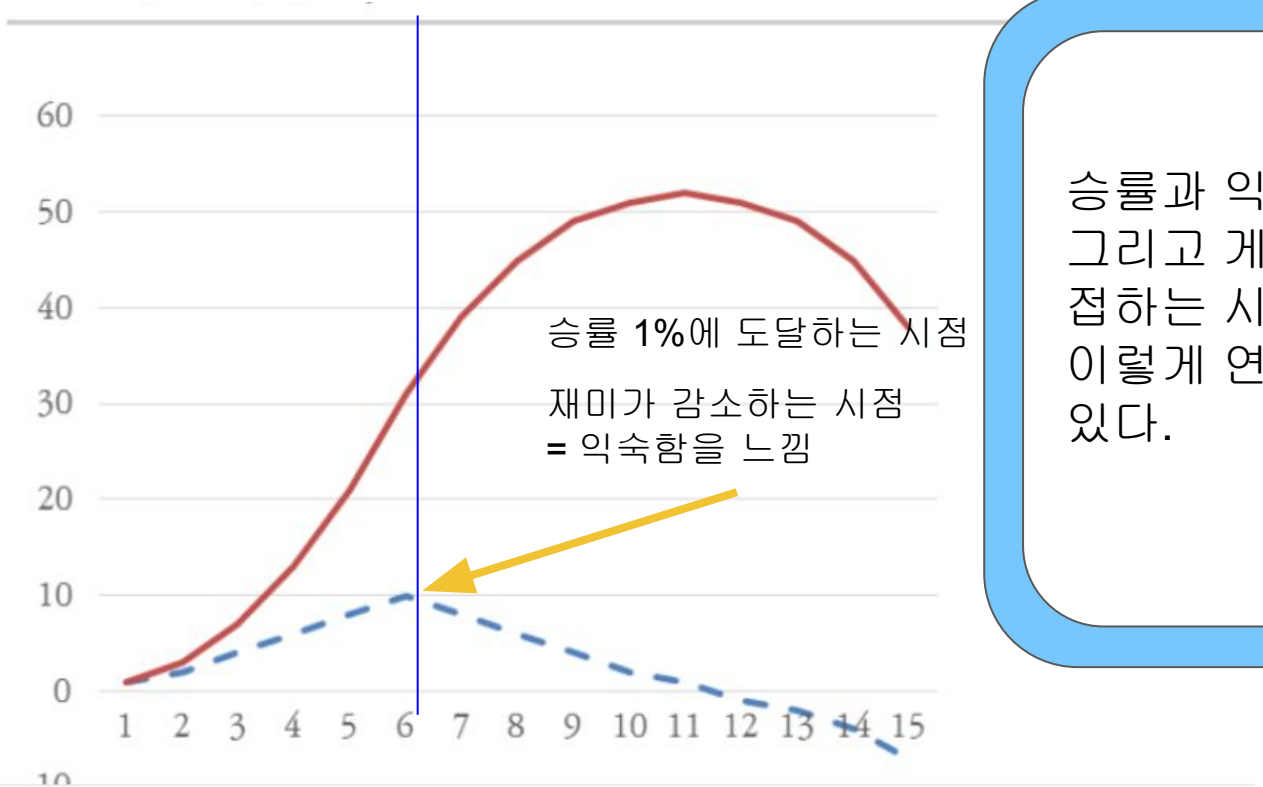
평균 점수가 높을수록 승률이 높아진다.

PDP



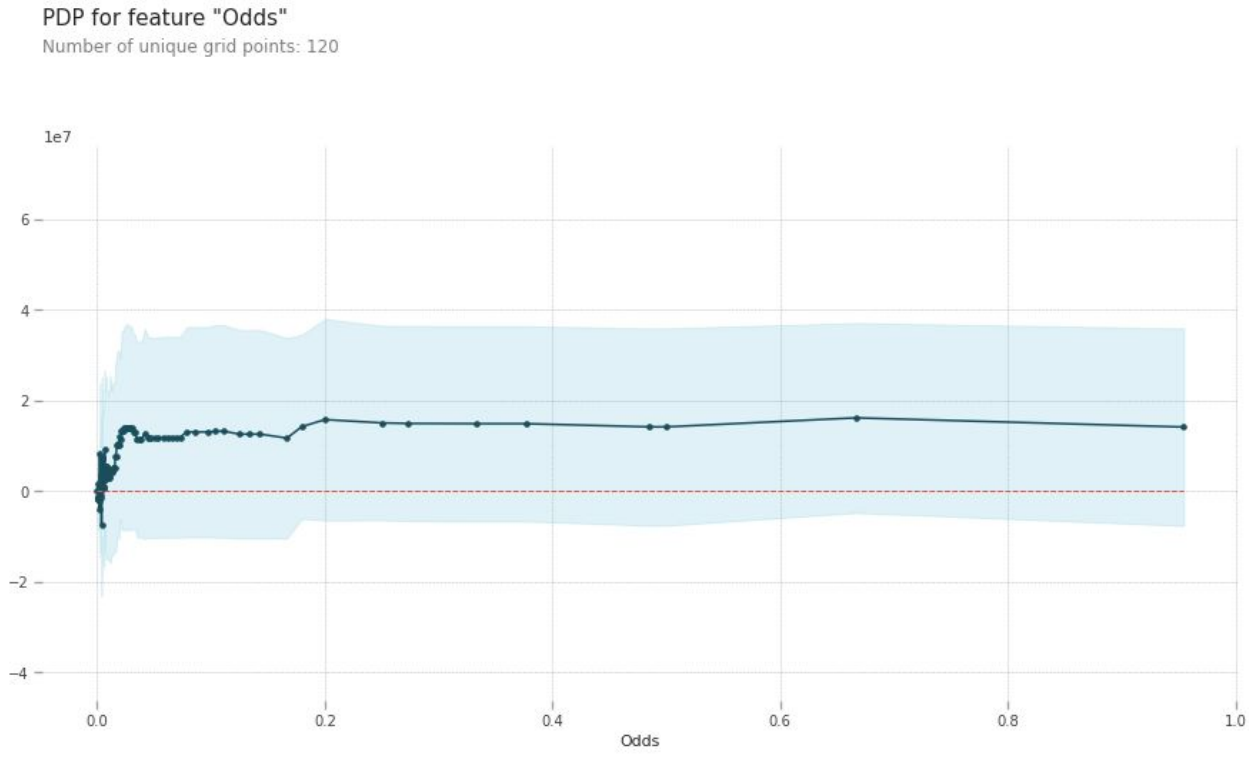
평균 점수가 높아질수록 승률은 올라가고, 이는 곧 게임에 익숙해졌기에 흥미가 떨어지며 이에 따라서 Playtime이 줄어든다고 볼 수도 있다.

한계효용체감의 법칙



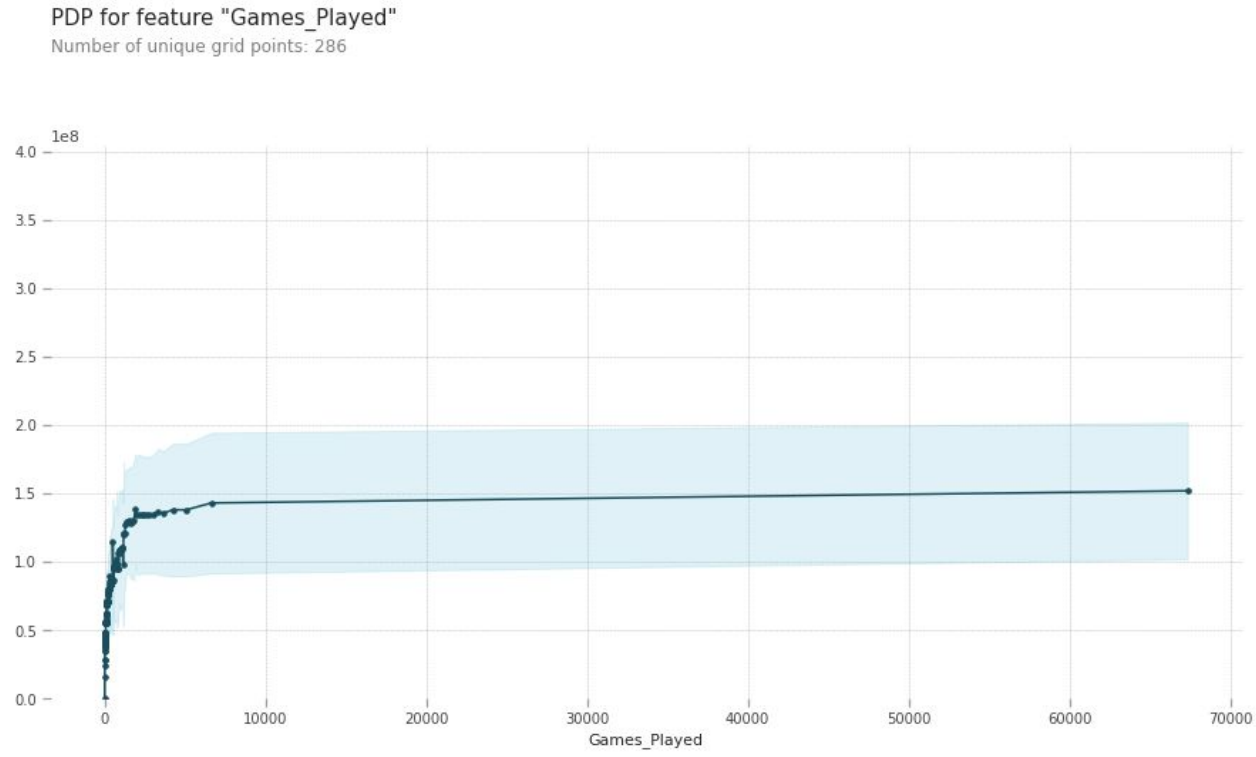
승률과 익숙함,
그리고 게임을
접하는 시간은
이렇게 연결될 수
있다.

PDP



승률(Odds)가 높아진다고 Playtime에 영향력이 커지진 않는다. 다만 일정 승률 구간의 경우에는 타겟이 감소하는 것이 확인된다.

PDP

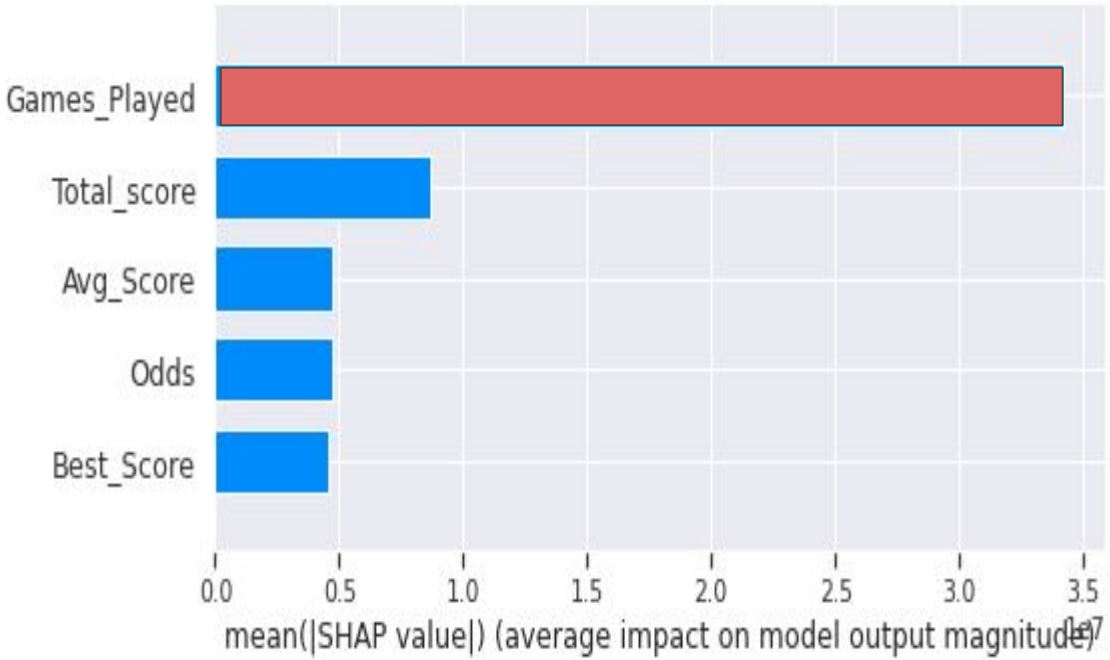


Games_Played(총 게임 판수)는 가장 영향력이 크다.

많이 할수록, 게임 시간이 늘어난다는 단순하면서도 당연한 논리가 존재하기 때문이다.

SHAP를 통한 분석

SHAP 특성 중요도



Games_Played(총 게임 판수)

압도적으로 중요도가 높은 특성이며, 앞의 분석에서도 확인되었습니다.

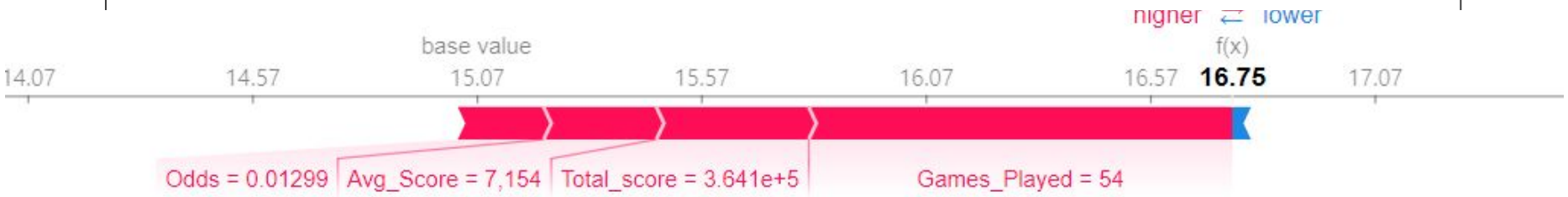
다른 특성들의 중요도는 상대적으로 낮지만 무시할 정도는 아니라고 판단합니다.

04

결론

일반적인 유저를 가정

모든 특성이 중간값에 해당한다고 가정하고(일반적 = 중간에 해당)
승률만 1%일 경우 Playtime을 예측합니다.



16.75라는 값을 변환해주면

즉, 약 217일과 17시간이 나옵니다.
그러나 이것은 단지 하나의 사례일 뿐입니다.

부트샘플링 기법의 활용

5%~95%

일반적인 유저

5%부터 10%, 15%... 95%
까지의 특성을 모두
랜덤하게 지정하여
10000개의 새로운
샘플을 만듭니다.

승률은 1%

목표 승률

특성은 모두 범위
내의 랜덤하게
부여했지만 승률은 1%
가 고정됩니다.

부트샘플링 활용 결과

	Total_score	Games_Played	Best_Score	Odds	Avg_Score
0	7679.2	20.0	1337028.0	0.012987	10081.0
1	549329.4	12.0	119302.8	0.012987	10081.0
2	62945.8	113.0	10108096.6	0.012987	26848.0
3	169978.4	282.0	10108096.6	0.012987	53.0
4	364123.0	149.0	3132.8	0.012987	13945.4
...	...	...	...	...	...
9995	7679.2	54.0	99.0	0.012987	13945.4
9996	62945.8	86.0	232406.6	0.012987	26848.0
9997	12672553.6	31.0	232406.6	0.012987	161125.8
9998	169978.4	113.0	509962.4	0.012987	3271.0
9999	62945.8	2.0	8930.0	0.012987	93140.8

10000 rows × 5 columns

모델 예측값의 평균을 일수로 변환하면

43일 6시간

새로운 게임이 만들어지는 이유

43일의 업데이트 주기를
감당해낼 개발진은 존재하지
않습니다.

게임에는 수명이 있고, DCSS와
비슷한 게임을 필요로 하는
유저층에겐 새로운 게임이
필요한 타이밍이라고
생각해볼 수 있습니다.

모델 학습의 한계

- 데이터의 특성 부족

-> 논리를 채우기 위한 요소가 부족했다

- 데이터의 신뢰성 부족

-> 아이디의 중복 생성이 가능, 이것을 확인할 방법 X

- 여러 기준에 대한 설정

-> 해당 기준의 신뢰 수준을 가늠하기 어려움 ...

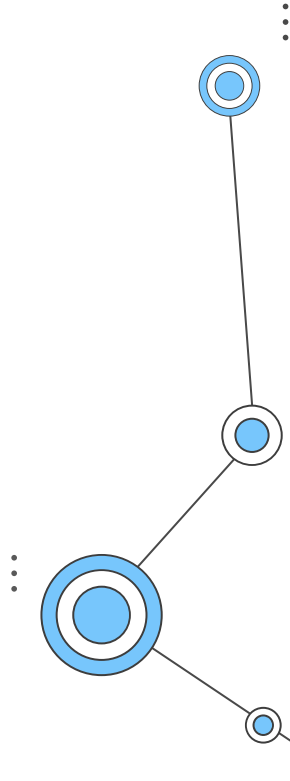
개선방안

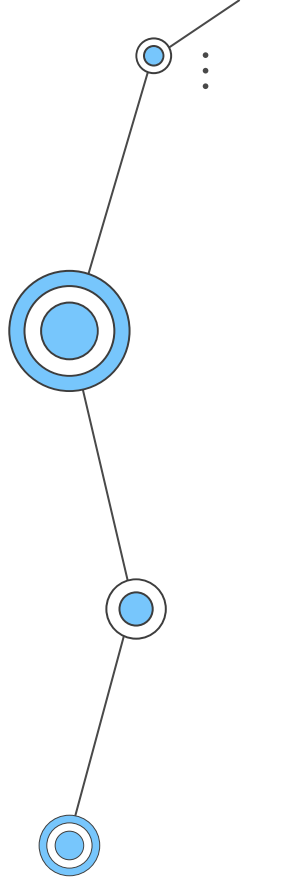
- 데이터의 특성 부족

-> 유저 마다의 게임 플레이 내용까지 특성으로 추가한 데이터를 만들어봅니다.

- 여러 기준에 대한 설정

-> 이러한 예측을 토대로 실무에 적용해보고, 결과를 통해 기준에 대한 조정을 진행해봅니다.





감사합니다

피드백은 강하고 진심이 담겨야 서로에게 좋다고 생각합니다

rudah69@gmail.com
박경모