

League of Legend Data Analysis

<AFK Player Detection in Match_data>

게임을 포기하는 플레이어 탐지 모델 생성

Task Definition

•AFK란? Away From KeyBoard 의 약자로, 게임을 중간에 완전히 포기하고 이길 의지가 없는 유저를 말합니다. 이번 프로젝트에서는 AFK를 통해 게임을 망치는 Troll player를 detect 합니다.

•아이템을 판매하거나, 특정 아이템들을 여러 개 구매하며, 게임을 포기하는 유저들을 아이템 선택을 이용하여 찾고, 그들의 특성을 (k/d/a, gold 획득량, 레벨) 들을 이용하여 detection 하는 모델을 만드는 것을 목표로 하였습니다.

솔랭

한달 전

패배

32분 4초

코그모

6 / 9 / 8

1.56:1 평점

ACE

레벨17

290 (9) CS

킬관여 58%

매치 평균

Diamond 2

제어 와드 2

이석희

맷 펠

Xle Xu W...

건모형안...

망나니 아...

칼바람은...

한뒤얼

WkdRoen...

xioyuki

휴먼강등

솔랭

한달 전

패배

21분 6초

코그모

1 / 6 / 3

0.67:1 평점

레벨13

161 (7.6) CS

킬관여 29%

매치 평균

Diamond 1

제어 와드 3

이석희

정글죽지...

hhy

jiu zhe

맷 펠

기모돌

물음표 받...

너구리 팬...

자아줄아...

마옹냐옹

종합

팀 분석

빌드

etc

패배 (블루팀)

티어

OP Score

KDA

피해량

와드

CS

아이템

13

이석희

Diamond 2

6.6

5th

0.67:1

1/6/3 (29%)

14,879

3

9 / 3

161

분당 7.6

11

정글죽지마...

Diamond 2

4.9

9th

0.60:1

1/5/2 (21%)

5,518

1

3 / 6

125

분당 5.9

10

hhy

Diamond 2

6.6

ACE

2.67:1

7/3/1 (57%)

11,843

5

6 / 4

109

분당 5.2

9

jiu zhe

Diamond 2

3.5

10th

0.67:1

0/3/2 (14%)

3,247

0

6 / 0

122

분당 5.8

10

맷 펠

Diamond 1

6.3

7th

1.50:1

5/6/4 (64%)

6,472

6

16 / 0

30

분당 1.4

0

2

1

Total Kill

14

23

Total Gold

34028

40837

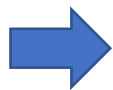
0

1

7

Troll Detection 분석 방법

특정 아이템 중복 구매 유저
→ Troll 유저일 가능성이
높다고 판단



특정 아이템 중복 구매
유저들의 k/d/a , gold, level
정보를 통해 Troll 유저의 특성
추출



앞의 Feature를 이용하여
아이템을 중복 구매하지 않은
Troll Detecting Model 생성
(NN 모델)

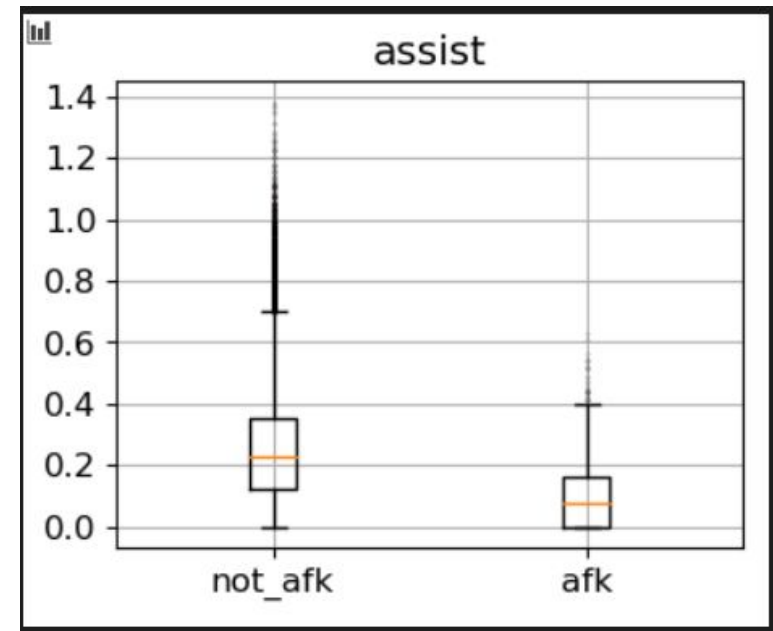
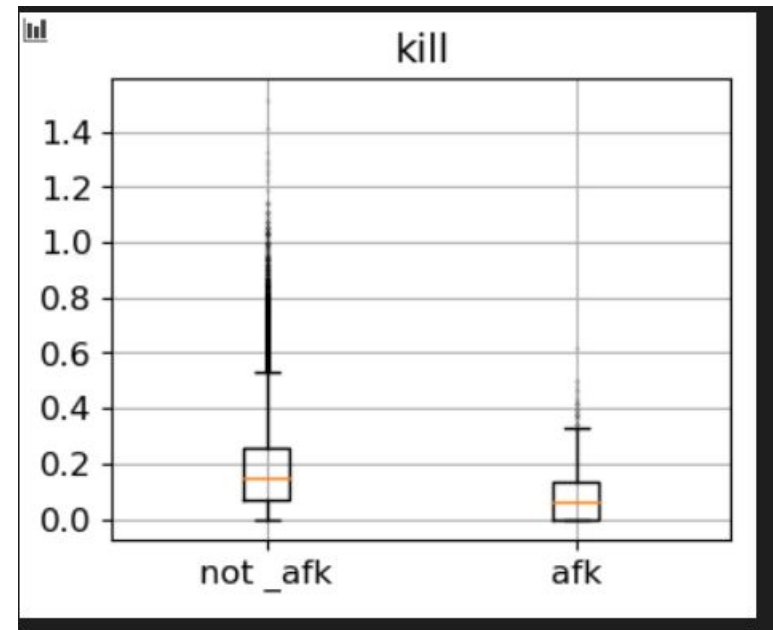
같은 아이템을 6개 사는 유저들

- 유저들이 게임 종료 시 같은 아이템을 6개 중복 구매한 아이템들을 뽑아, 정렬해보았습니다.
- 원기회복 구슬을 구매한 2230명과, 요정의 부적을 구매한 914명, 부러진 초시계 511명이 눈에 띕니다.
- 암흑의 인장, 사파이어 수정, 루비 수정, 단검 등 흔히 게임을 포기하는 유저들이 많이 구매하는 아이템들을 볼 수 있었습니다.
- 흔히 트롤의 상징인 여신의 눈물이나 신발은 중복구매가 불가해졌기에, 구매내역에서 볼 수 없었습니다.

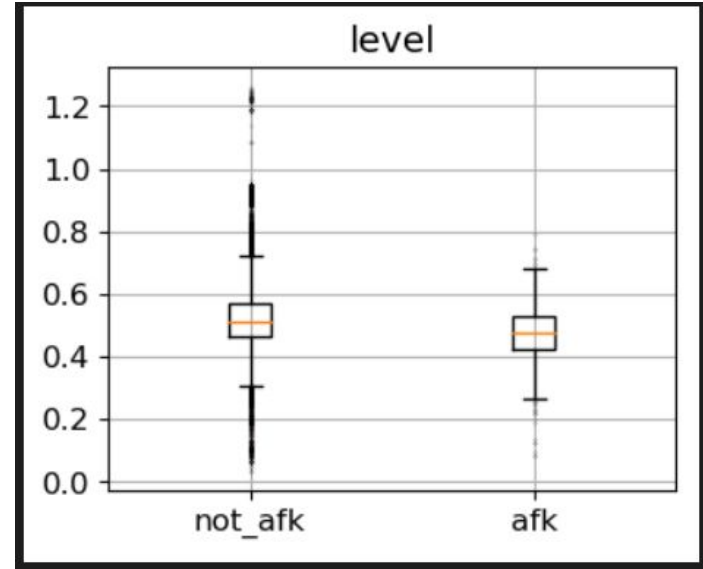
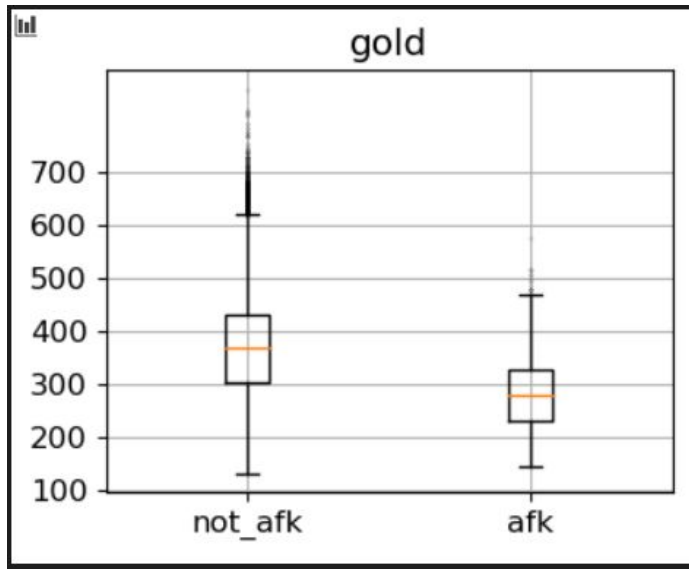
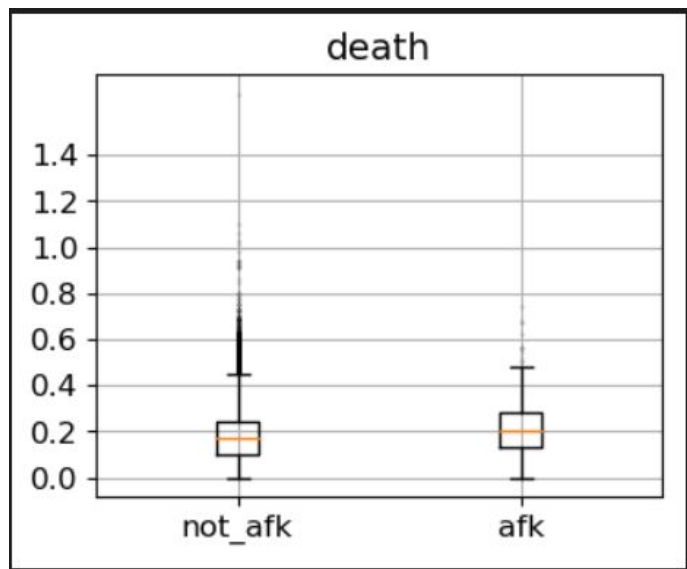
Rejuvenation Bead 2230
Faerie Charm 914
Broken Stopwatch 511
Dark Seal 418
Sapphire Crystal 334
Stopwatch 241
Dagger 238
Ruby Crystal 233
Doran's Ring 139
Doran's Blade 120
Bami's Cinder 115
Cull 115
Long Sword 104
Kindlegem 102
Chalice of Harmony 76
Cloth Armor 75
Kircheis Shard 47
Crystalline Bracer 46
Executioner's Calling 39
Amplifying Tome 36
Aether Wisp 26
Mejai's Soulstealer 25
Null-Magic Mantle 24
Giant's Belt 24
Doran's Shield 24

AFK 유저와 일반 유저의 통계적 차이 비교

- 원기회복의 구슬, 요정의 부적, 사파이어 수정, 루비 수정 을 6개 구매한 유저들과, 초시계, 부러진 초시계를 2개 이상 구매한 유저들을 afk 유저로 판단
- Kill , Death, Assist , Level , Gold 값을 Feature 로 , 해당 유저들의 Feature 데이터들 시각화

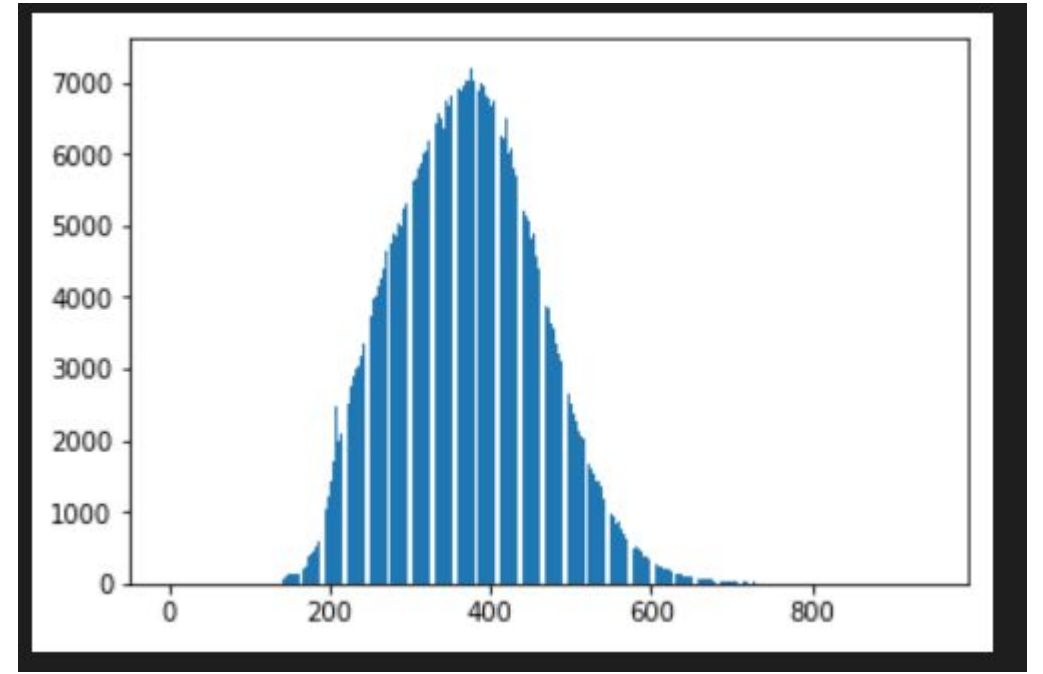


AFK 유저와 일반 유저의 통계적 차이 비교2



챔피언별 Data Standard Normalization

- 전체 데이터로 트롤 플레이어를 구분하기에는 각 챔피언별 특징이 크게 작용하여 AFK Detecting Feature로 사용하기에 적합하지 않을 것으로 예상됩니다.
- 각각의 5개의 피처에 챔피언별로 평균과 분산을 구해 Standard Normalization 을 진행하였습니다.
- 데이터가 더욱 충분하다면, 포지션 별 - 챔피언 별 통계치를 이용한다면 더욱 좋은 결과를 얻을 수 있을 것으로 보입니다.



게임 종료 시 분당 골드획득량에 따른
유저수 그래프입니다.
정규분포를 따르는 것으로 보입니다.

AFK 유저와 일반 유저의 통계적 차이 비교

- 챔피언별로 피쳐들을 표준 정규화 진행 후
- 데이터 시각화 및 T - test 수행

데이터 표준정규화 전
T - test 결과

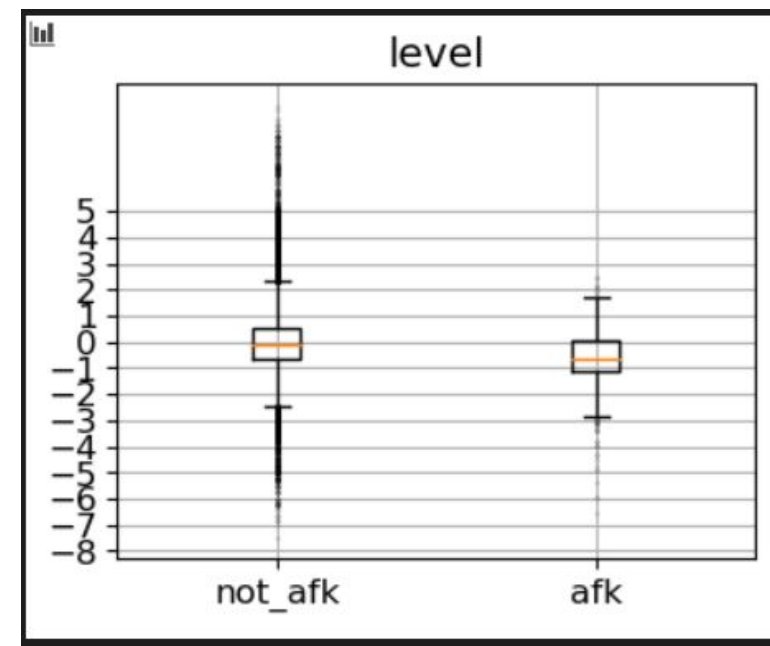
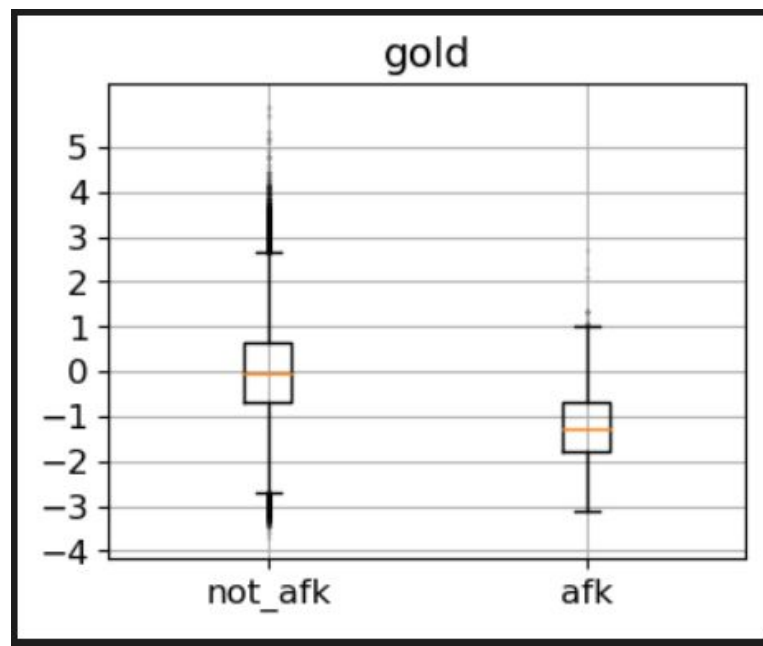
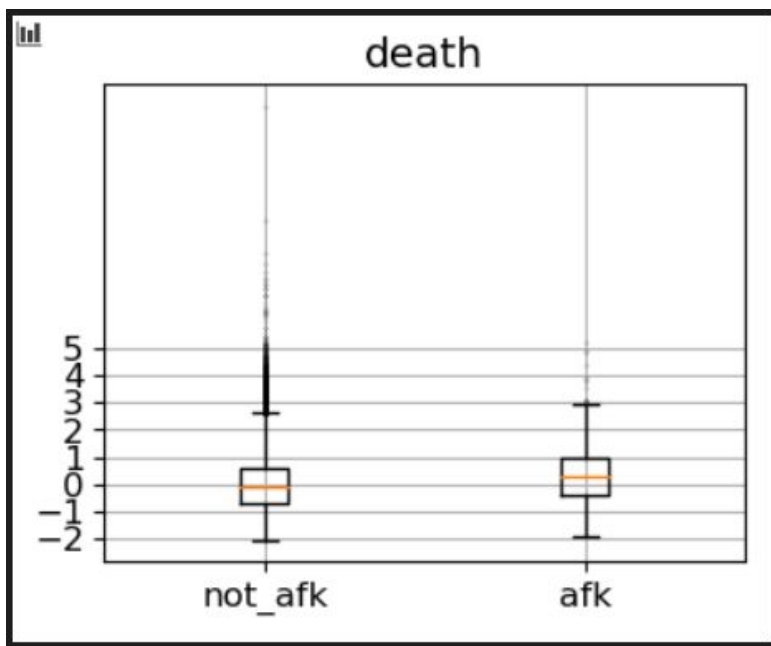
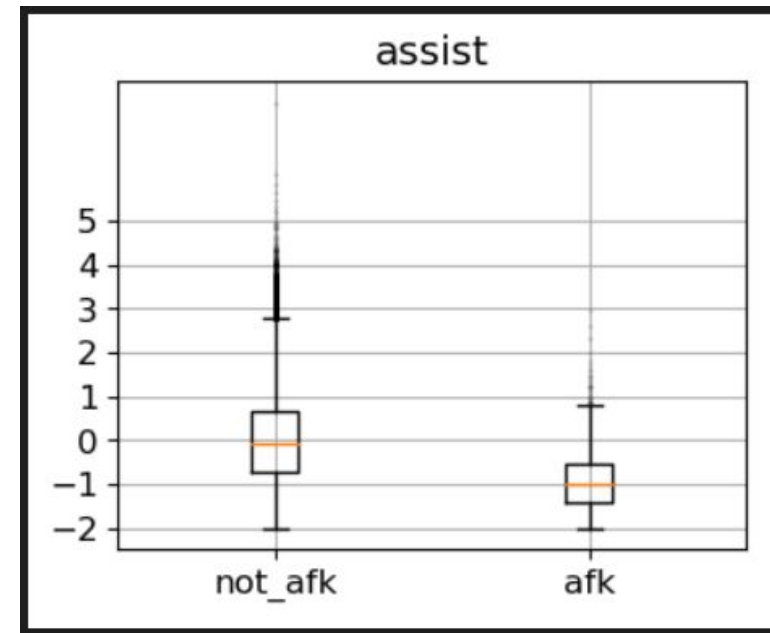
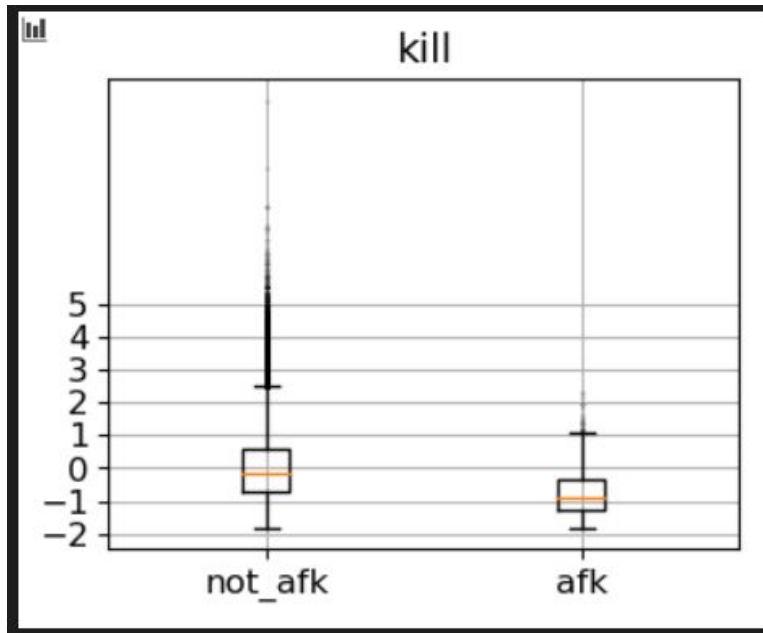
```
kill Ttest_indResult(statistic=30.06008579959953, pvalue=3.3658330351613374e-146)
assist Ttest_indResult(statistic=43.29711374279361, pvalue=4.699904044994977e-242)
death Ttest_indResult(statistic=-10.905827458048604, pvalue=2.2357696428937928e-26)
gold Ttest_indResult(statistic=38.62142327982886, pvalue=5.947552904795588e-208)
level Ttest_indResult(statistic=16.82463944014859, pvalue=9.396355808774265e-57)
```



데이터 표준정규화 후
T - test 결과

```
kill Ttest_indResult(statistic=33.31392491572517, pvalue=8.58250752238684e-170)
assist Ttest_indResult(statistic=41.1409268659045, pvalue=2.738716219255026e-226)
death Ttest_indResult(statistic=-10.598853957274113, pvalue=4.5553456290493745e-25)
gold Ttest_indResult(statistic=44.88380253216268, pvalue=5.193520136771506e-252)
level Ttest_indResult(statistic=20.72909519167909, pvalue=6.145569357307335e-81)
```


표준 정규화 후 데이터



DataSet 생성

```
neg, pos = np.bincount(y)
total = neg + pos
print('Examples:\n    Total: {} \n    Positive: {} ({:.2f}% of total)\n'.format(
    total, pos, 100 * pos / total))

Examples:
    Total: 57354
    Positive: 5765 (10.05% of total)
```

- AFK 데이터 5천건과 샘플링을 통해 추출한 5만2천건의 데이터를 합친 후, Shuffle하여 데이터셋을 생성하였습니다.
- Positive 데이터가 10퍼센트인 데이터 셋으로,
- inbalance problem이 발생 할 수 있습니다.

NN 모델

```
model = keras.Sequential()
model.add(tf.keras.layers.Dense(5,input_shape=(5,),activation='relu'))
model.add(tf.keras.layers.Dense(5,activation='relu'))
model.add(tf.keras.layers.Dense(10,activation='relu'))
model.add(tf.keras.layers.Dense(1,activation='sigmoid'))
model.compile(optimizer="adam", loss="binary_crossentropy",metrics=['binary_accuracy',tf.keras.metrics.Recall(name='Recall'),
tf.keras.metrics.Precision(name='Precision')])

model.fit(x_train,y_train,epochs=20,batch_size=50,verbose=1)
```

- 위는 Tensorflow를 통해 만든 NN Model입니다.
- NN 모델에 넣어 accuracy , Recall , Presicion 을 확인해보았습니다.

Train Result

```
model.fit(x_train,y_train,epochs=20,batch_size=50,verbose=1)
Epoch 10/20
803/803 [=====] - 2s 2ms/step - Recall: 0.2225 - binary_accuracy: 0.9059 - Precision: 0.5860 - loss: 0.2486
Epoch 11/20
803/803 [=====] - 2s 2ms/step - Recall: 0.2202 - binary_accuracy: 0.9062 - Precision: 0.5918 - loss: 0.2485
Epoch 12/20
803/803 [=====] - 2s 2ms/step - Recall: 0.2185 - binary_accuracy: 0.9059 - Precision: 0.5875 - loss: 0.2481
Epoch 13/20
803/803 [=====] - 2s 2ms/step - Recall: 0.2249 - binary_accuracy: 0.9063 - Precision: 0.5910 - loss: 0.2482
Epoch 14/20
803/803 [=====] - 2s 3ms/step - Recall: 0.2193 - binary_accuracy: 0.9062 - Precision: 0.5919 - loss: 0.2479
Epoch 15/20
803/803 [=====] - 2s 2ms/step - Recall: 0.2230 - binary_accuracy: 0.9068 - Precision: 0.5999 - loss: 0.2480
Epoch 16/20
803/803 [=====] - 2s 2ms/step - Recall: 0.2217 - binary_accuracy: 0.9064 - Precision: 0.5942 - loss: 0.2478
Epoch 17/20
803/803 [=====] - 1s 2ms/step - Recall: 0.2272 - binary_accuracy: 0.9068 - Precision: 0.5973 - loss: 0.2477
Epoch 18/20
803/803 [=====] - 2s 2ms/step - Recall: 0.2178 - binary_accuracy: 0.9059 - Precision: 0.5878 - loss: 0.2477
Epoch 19/20
803/803 [=====] - 2s 2ms/step - Recall: 0.2225 - binary_accuracy: 0.9062 - Precision: 0.5911 - loss: 0.2477
Epoch 20/20
803/803 [=====] - 2s 2ms/step - Recall: 0.2301 - binary_accuracy: 0.9065 - Precision: 0.5920 - loss: 0.2477
<tensorflow.python.keras.callbacks.History at 0x1b6b7a55f98>
```

- Train Result

Test Result

```
▶ ▶ M↓  
model.evaluate(x_test,y_test,batch_size=50)  
241/241 [=====] - 0s 2ms/step - binary_accuracy: 0.8785 - Precision: 0.6710 - loss: 0.3048 - Recall: 0.3234  
[0.3048098683357239, 0.8784558176994324, 0.32344552874565125, 0.67100590467453]
```

- 비교적 Recall이 낮고, Precision이 높은 것으로 볼 때,
- Inbalance Problem이 나타나서 대부분의 데이터를 0으로 예측하고 있음을 짐작 할 수 있습니다.

Weight Balancing

```
weight_for_0 = (1 / neg)*(total)/2.0
weight_for_1 = (1 / pos)*(total)/2.0

class_weight = {0: weight_for_0, 1: weight_for_1}

print('Weight for class 0: {:.2f}'.format(weight_for_0))
print('Weight for class 1: {:.2f}'.format(weight_for_1))

Weight for class 0: 0.56
Weight for class 1: 4.97
```

```
model.fit(x_train,y_train,epochs=20,batch_size=50,verbose=1,class_weight=class_weight)
-----
Epoch 10/20
803/803 [=====] - 2s 2ms/step - Recall: 0.7644 - binary_accuracy: 0.7629 - Precision: 0.2650 - loss: 0.4953
Epoch 11/20
803/803 [=====] - 2s 2ms/step - Recall: 0.7637 - binary_accuracy: 0.7646 - Precision: 0.2665 - loss: 0.4948
Epoch 12/20
803/803 [=====] - 2s 2ms/step - Recall: 0.7602 - binary_accuracy: 0.7664 - Precision: 0.2676 - loss: 0.4940
```

Imbalance Problem 을 해결하기 위해 class_weight을 추가하였습니다.











































Weight Balancing Result

```
model.evaluate(x_test,y_test,batch_size=50)
```

```
345/345 [=====] - 1s 2ms/step - Recall: 0.7622 - binary_accuracy: 0.7615 - Precision: 0.2624 - loss: 0.5027
```


















































- Accuracy 는 감소하고, Precision 이 줄면서 Recall 이 증가했습니다.
- AFK 유저 중, 게임을 포기하였으나 아이템을 팔고 바꾸는 유저의 비율이 많지 않을 것이라 생각했고, Precision 이 낮은 것은 그러한 이유일 것으로 예측됩니다.

Result Validation 결과 검증 Precison

	kill	assist	death	gold	level	troll	championid	item0	item1	item2	item3	item4	item5
23958	-0.574912	0.155743	1.355545	-0.934616	-0.087305	0	142						
23962	-0.995607	-1.687595	-1.392441	-2.026246	-4.956815	0	43						
23963	-1.400975	-1.439424	0.098945	-1.737542	1.199684	0	69						
23965	0.459464	-1.785931	3.160320	-0.592500	0.312200	0	98						
23984	-0.943732	-0.391564	1.318360	-1.486425	0.386556	0	69						
23985	-1.369390	-1.589085	-0.515048	-2.109652	-1.644368	0	81						
23988	-0.450908	-0.490896	1.602401	-0.942134	-0.797476	0	17						
23990	-0.938473	-0.515920	1.180931	-1.693229	-0.236060	0	84						
23991	-1.324016	-1.587531	-1.791949	-2.344058	-2.970216	0	523						
23999	-1.565611	-1.723524	-0.936741	-2.806229	-2.711272	0	5						























































- 실제 아이템으로 afk 플레이가 아니라고 구분한 데이터 중 , 모델이 afk 플레이로 판단한 데이터들입니다.
- 빨간 박스에 보이는 것처럼 아이템을 전부 혹은 거의 다 판매한 유저들이 보입니다.

Result Validation 결과 검증 Precision

	kill	assist	death	gold	level	troll	championid	item0	item1	item2	item3	item4	item5
23958	-0.574912	0.155743	1.355545	-0.934616	-0.087305	0	142						
23962	-0.995607	-1.687595	-1.392441	-2.026246	-4.956815	0	43						
23963	-1.400975	-1.439424	0.098945	-1.737542	1.199684	0	69						
23965	0.459464	-1.785931	3.160320	-0.592500	0.312200	0	98						
23984	-0.943732	-0.391564	1.318360	-1.486425	0.386556	0	69						
23985	-1.369390	-1.589085	-0.515048	-2.109652	-1.644368	0	81						
23988	-0.450908	-0.490896	1.602401	-0.942134	-0.797476	0	17						
23990	-0.938473	-0.515920	1.180931	-1.693229	-0.236060	0	84						
23991	-1.324016	-1.587531	-1.791949	-2.344058	-2.970216	0	523						
23999	-1.565611	-1.723524	-0.936741	-2.806229	-2.711272	0	5						























































- 박스에 보이는 것처럼, 크게 의미 없는 아이템들로 아이템을 산 모습을 볼 수 있습니다.
- 물론 특정한 경우엔, 올바른 플레이를 하는 유저들도 구매했을 법한 아이템 조합이지만 킬, 데스, 어시스트, 골드, 레벨 지표를 살펴보면 제대로 게임을 수행하지 않다는 것을 알 수 있습니다.

Result Validation 결과 검증 Recall

	kill	assist	death	gold	level	troll	championid	item0	item1	item2	item3	item4	item5
23892	1.188684	3.013323	0.984122	0.225111	-1.050882	1	91						
24053	1.207006	-1.357017	0.320586	0.563737	0.698133	1	80						
24077	-0.323467	-0.405944	1.088179	-0.559149	-1.033992	1	517						
24098	0.063402	-0.181606	-1.037052	-0.037530	-0.128689	1	80						
24165	-0.559930	1.864688	0.200992	-0.409071	0.102344	1	11						
24213	-0.488163	0.026460	0.229208	-0.439995	-0.538801	1	80						
24231	-1.278346	-0.668653	-0.751727	-0.191620	0.693109	1	120						
24255	0.252187	-0.596765	-0.414108	0.403689	-0.678331	1	122						
24311	0.217005	1.992677	0.093604	-0.024714	-1.220291	1	103						

- 실제 아이템으로 afk 플레이어라고 구분한 데이터 중 , 모델이 afk 플레이가 아니라고 판단한 데이터들입니다.
- 빨간 박스에 보이는 것처럼, kda가 준수하고, Level은 밀리는 유저들입니다. 게임 초중반에 잘하다가 어느 순간 게임을 포기한 유저들입니다. 대표적인 afk 유저이지만, feature값이 좋아 모델이 확인하기 어렵습니다.

















































Result Validation 결과 검증 Recall

	kill	assist	death	gold	level	troll	championid	item0	item1	item2	item3	item4	item5
23892	1.188684	3.013323	0.984122	0.225111	-1.050882	1	91						
24053	1.207006	-1.357017	0.320586	0.563737	0.698133	1	80						
24077	-0.323467	-0.405944	1.088179	-0.559149	-1.033992	1	517						
24098	0.063402	-0.181606	-1.037052	-0.037530	-0.128689	1	80						
24165	-0.559930	1.864688	0.200992	-0.409071	0.102344	1	11						
24213	-0.488163	0.026460	0.229208	-0.439995	-0.538801	1	80						
24231	-1.278346	-0.668653	-0.751727	-0.191620	0.693109	1	120						
24255	0.252187	-0.596765	-0.414108	0.403689	-0.678331	1	122						
24311	0.217005	1.992677	0.093604	-0.024714	-1.220291	1	103						

- 피쳐들의 값만으로도, afk유저임을 거의 확신 할 수 있는 데이터이며, 모델이 잘못 잡아낸 경우입니다.

특정 유저가 AFK플레이를 할 확률은?

- 아래는 아이템을 판매하고 게임을 포기한 횟수가 가장 많은 유저의 게임들입니다.
- 빨간 박스의 게임과 같이 게임을 포기하지 않고 모든 지표가 좋게 플레이을 한 경우에도, 게임이 끝나기 직전에 아이템을 바꾼것 으로 보이는 게임들이 종종 있습니다.
- 위에서 학습한 모델을 사용하여 이러한 게임들을 제외하고, 유저들이 AFK 플레이 한 비율을 계산하여 서비스에 반영 할 수 있을 것입니다.

	kill	assist	death	gold	level	troll	championid	summoner_id	item0	item1	item2	item3	item4	item5
7515	2.284572	1.678524	-1.280781	2.707909	1.798161	1	82	Dw-MTTbD78UU4xosHihNXSRXgt4nUqfHz-FA7KjutKp1BoM						
8295	-0.975253	0.684435	-0.954245	0.303592	1.057325	1	350	Dw-MTTbD78UU4xosHihNXSRXgt4nUqfHz-FA7KjutKp1BoM						
34890	-1.102568	-1.590245	-0.947922	-2.030715	-3.344306	1	81	Dw-MTTbD78UU4xosHihNXSRXgt4nUqfHz-FA7KjutKp1BoM						
39785	0.616850	-1.535883	2.556752	-0.262227	-0.389238	1	235	Dw-MTTbD78UU4xosHihNXSRXgt4nUqfHz-FA7KjutKp1BoM						
40275	-1.379865	-1.646646	0.362283	-3.327912	-6.278903	0	51	Dw-MTTbD78UU4xosHihNXSRXgt4nUqfHz-FA7KjutKp1BoM						
43938	-0.222396	-0.734838	-0.057476	-1.209471	-1.706031	1	81	Dw-MTTbD78UU4xosHihNXSRXgt4nUqfHz-FA7KjutKp1BoM						
50828	-1.365494	-1.590245	-0.789649	-2.737265	-3.771532	1	81	Dw-MTTbD78UU4xosHihNXSRXgt4nUqfHz-FA7KjutKp1BoM						
52933	-1.165412	-1.021191	-0.623082	-1.601361	-1.163682	1	235	Dw-MTTbD78UU4xosHihNXSRXgt4nUqfHz-FA7KjutKp1BoM						
64090	-0.228466	0.404265	-1.433933	0.347551	0.625687	0	350	Dw-MTTbD78UU4xosHihNXSRXgt4nUqfHz-FA7KjutKp1BoM						

AFK 유저 탐지 서비스

- 오른쪽과 같이 AFK 플레이 비율을 opgg 사이트의 “여러명의 소환사 이름으로 요약 검색 기능”에 추가적으로 넣어주는 서비스에 활용 할 수 있을 것으로 보입니다.
- 최근 플레이칸에는 AFK 플레이에 기존의 ‘ACE’, ‘MVP’처럼 ‘AFK’마크를 달아주어 AFK 플레이를 쉽게 알아 볼 수 있도록 해주고,
- 연패 연승을 표시해주는 부분에 최근 플레이 중 AFK 비율을 표시해주는 방향으로 추가가 가능합니다.
- 이렇게 솔랭에서 매칭된 팀원들이 AFK 유저라는 것을 쉽게 알아 볼 수 있게 해주면, 플레이어들의 닷지 / 인게임에서의 판단에 큰 도움을 줄 수 있을 것이라 생각합니다.

S2020 Platinum

최근 업데이트: 8일 전

너를보고웃다
Platinum 4 (0 LP)

609승 617패 50%

3연패중 AFK 20%

최근 플레이

80% W/R 25%	20% W/R 0%
4 / 6 / 5 A	8일 전
14 / 8 / 7	8일 전
6 / 7 / 6	한달 전
11 / 1 / 6 M	한달 전
0 / 3 / 0 AFK	한달 전
9 / 8 / 11	한달 전
3 / 6 / 4	한달 전
0 / 8 / 0 AFK	한달 전
7 / 7 / 8 A	한달 전
5 / 1 / 6	한달 전

2.67:1 KDA 716 54%