League of Legend(LOL) 승부 예측 모델

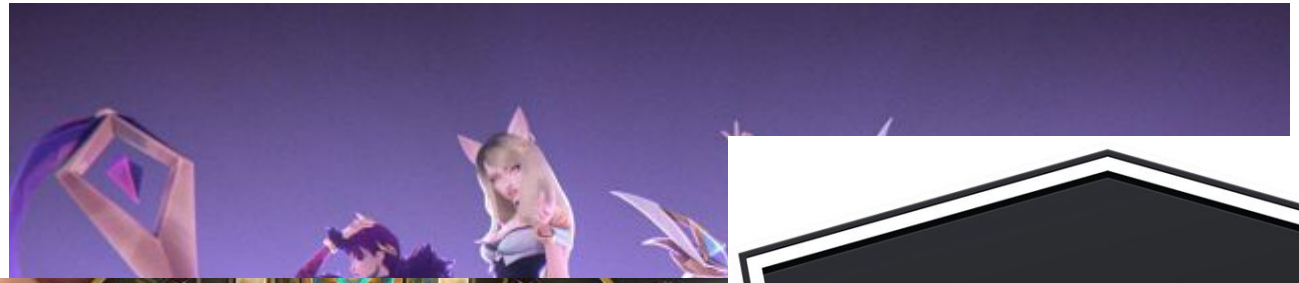
데이터 분석 및 모델 예측

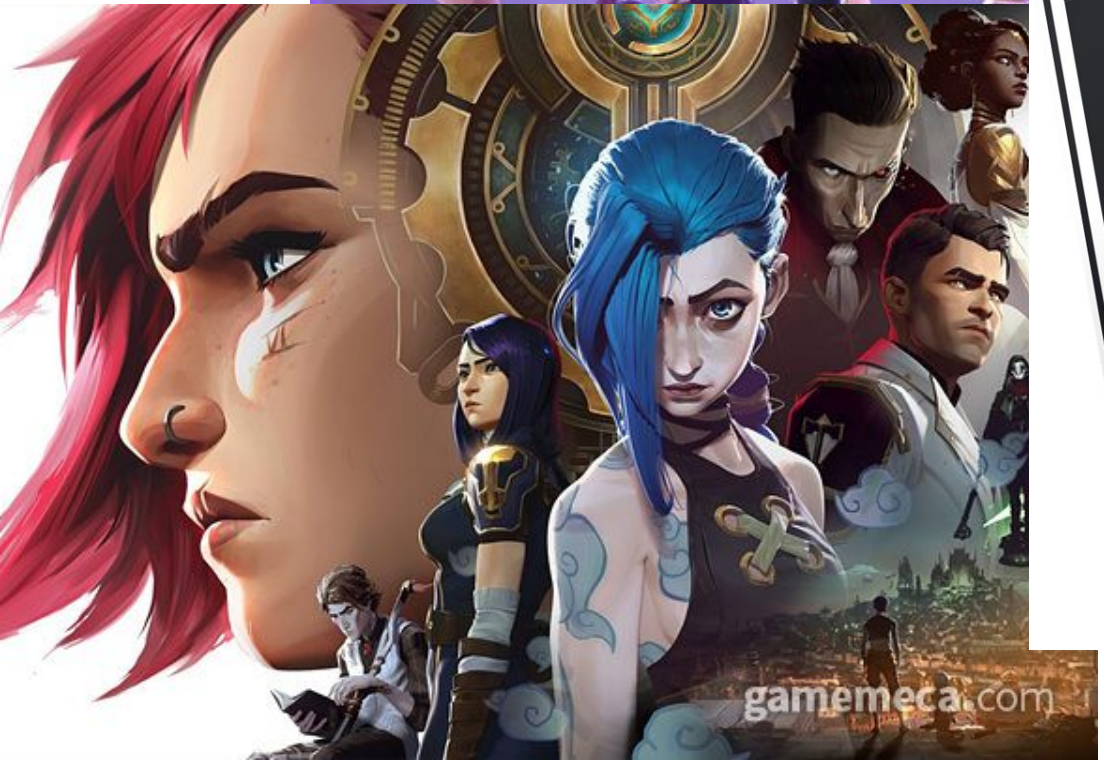
목차

1. 개발 목표
2. 데이터 분석 및 처리
3. 결과 예측(모델링)

개발 목표


LEAGUE of LEGENDS





LCK
ACADEMY
SERIES

우리 마음속 첫번째 금융

'LoL', 아시안게임 이어 올림픽까지 정식 종목으로 채택되나

2022년 항저우 아시안게임에 정식 종목으로 채택된 e스포츠의 대장급 게임인 '리그 오브 레전드'(이하 LoL)가 올림픽 정식 종목으로의 채택을 위해 국제올림픽조직위원회(IOC)의 고위층과 논의 중인 것으로 알려졌다.

e스포츠, 2022년 항저우 아시안게임 정식종목 채택

질병코드 등록 탓 위축된 게임 위상 상승 반영

지난 4월 회의 땐 '탈락'... 코로나로 e스포츠 위상 변화

2018년 시범종목 이어 2022년 정식종목

전문가 "e스포츠가 산업적으로 제도권에 들어온 것"

국산 게임 종목 없는 건 한계… "게임업계 분발해야"

지난 2018년 자카르타-팔렘방 아시안게임에서 시범종목으로 치러진 e스포츠가 오는 2022년 항저우 아시안게임의 정식종목에 편입된다.

데이터 분석

데이터 출처
https://www.kaggle.com/datasets/datasnaek/league-of-legends

- Game ID : 게임 계정
- Creation Time : 계정 생성일
- Game Duration : 게임 경기 시간 (second 단위)
- Season ID : 게임시즌 (9시즌 데이터)

- First : 선취팀 (1 = Bl
- T1, T2 : Blue팀 , Red
  - Baron : 내셔 남작
  - dragon : 용
  - tower : 타워
  - blood : 첫킬
  - inhibitor : 억제기
  - Rift Herald : 전령

```
# target 1 = Blue 2 = Red
game_winner['winner'].value_counts()
```

```
1    25441
2    24821
Name: winner, dtype: int64
```

```
# target 1 = Blue 2 = Red
game_winner['winner'].value_counts()
```

```
1    25441
2    24821
Name: winner, dtype: int64
```

	gameId
0	3326086514
1	3229566029
2	3327363504
3	3326856598
4	3330080762
...	...
51485	3308904636
51486	3215685759
51487	3322765040
51488	3256675373
51489	3317333020

	creationTime
0	1504279457970
1	1497848803862
2	1504360103310
3	1504348503996
4	1504554410899
...	...
51485	1503076540231
51486	1496957179355
51487	1504029863961
51488	1499562036246
51489	1503612754059

	seasonId
0	9
1	9
2	9
3	9
4	9
...	...
51485	9
51486	9
51487	9
51488	9
51489	9

```
# 비정상 게임
game_winner[(game_winner['gameDuration']<700) & ((game_winner['t1_towerKills']<11)|(game_winner['t2_towerKills']<11))].sort_values(by='gameDuration')
```

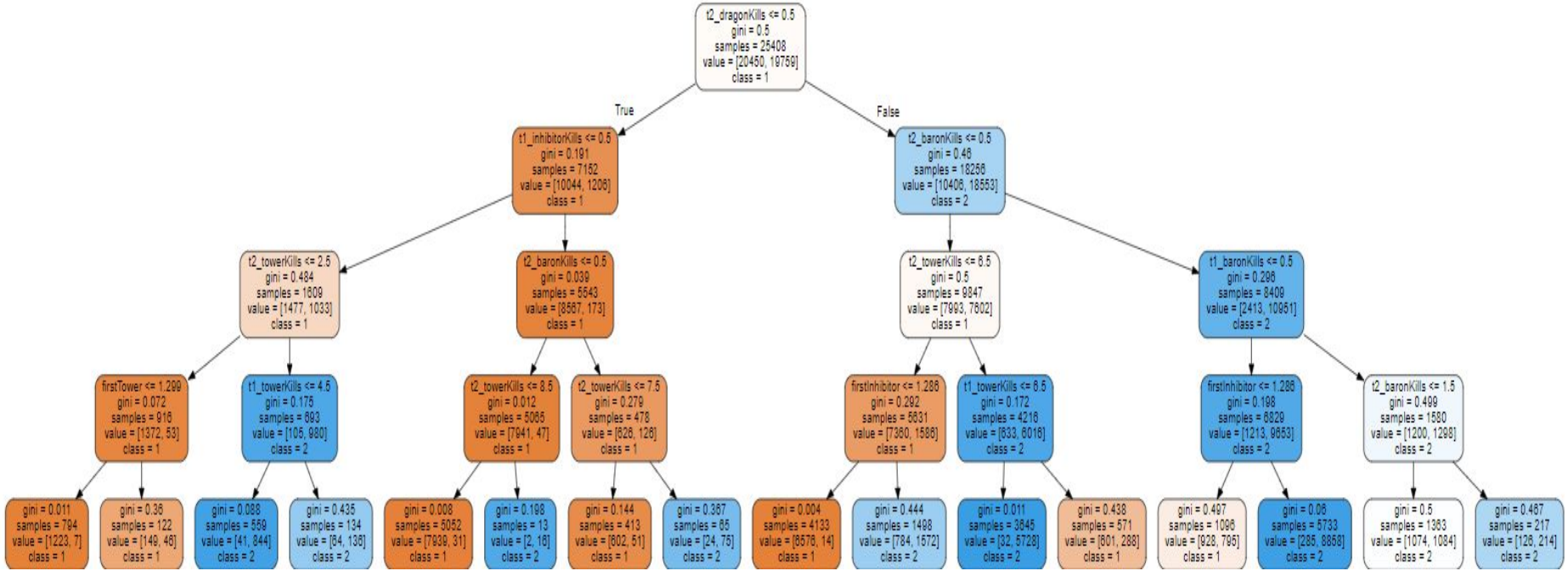
	gameDuration	winner	firstBlood	firstTower	firstInhibitor	firstBaron	firstDragon	firstRiftHerald	t1_towerKills	t1_inhibitorKills	t1_baronKills
5405	190	1	Red	None	None	None	None	None	0	0	0
46157	190	2	Red	None	None	None	None	None	0	0	0
34226	191	2	None	None	None	None	None	None	0	0	0
49702	191	1	None	None	None	None	None	None	0	0	0
7447	191	2	Red	None	None	None	None	None	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
37863	679	2	Red	Red	Red	None	None	None	0	0	0
15417	680	1	Red	Blue	Blue	None	None	None	8	1	0
16497	681	1	Blue	Blue	Blue	None	None	None	6	1	0
18734	684	1	Blue	Blue	Blue	None	None	None	5	1	0
43720	688	1	Blue	Blue	Blue	None	Blue	None	7	1	0

1222 rows × 18 columns

```
X_train['gametime'].value_counts()
```

```
30          16876
20          16528
40           4313
10           1930
50            526
overhour       36
```

결과 예측(모델링)


t2_dragonKills <= 0.5
gini = 0.5
samples = 25408
value = [20450, 19759]
class = 1
True
False
t1_inhibitorKills <= 0.5
gini = 0.191
samples = 7152
value = [10044, 1206]
class = 1
t2_baronKills <= 0.5
gini = 0.46
samples = 18256
value = [10406, 18553]
class = 2
t2_towerKills <= 2.6
gini = 0.484
samples = 1609
value = [1477, 1033]
class = 1
t2_baronKills <= 0.5
gini = 0.039
samples = 5543
value = [8567, 173]
class = 1
t2_towerKills <= 6.5
gini = 0.5
samples = 9847
value = [7993, 7602]
class = 1
t1_baronKills <= 0.5
gini = 0.296
samples = 8409
value = [2413, 10951]
class = 2
firstTower <= 1.299
gini = 0.072
samples = 916
value = [1372, 53]
class = 1
t1_towerKills <= 4.5
gini = 0.175
samples = 693
value = [105, 980]
class = 2
t2_towerKills <= 8.5
gini = 0.012
samples = 5065
value = [7941, 47]
class = 1
t2_towerKills <= 7.5
gini = 0.279
samples = 478
value = [626, 126]
class = 1
firstInhibitor <= 1.286
gini = 0.292
samples = 5631
value = [7360, 1586]
class = 1
t1_towerKills <= 8.5
gini = 0.172
samples = 4218
value = [633, 6016]
class = 2
firstInhibitor <= 1.286
gini = 0.198
samples = 6829
value = [1213, 9653]
class = 2
t2_baronKills <= 1.5
gini = 0.499
samples = 1580
value = [1200, 1298]
class = 2
gini = 0.011
samples = 794
value = [1223, 7]
class = 1
gini = 0.36
samples = 122
value = [149, 46]
class = 1
gini = 0.088
samples = 559
value = [41, 844]
class = 2
gini = 0.435
samples = 134
value = [64, 136]
class = 2
gini = 0.008
samples = 5052
value = [7939, 31]
class = 1
gini = 0.198
samples = 13
value = [2, 16]
class = 2
gini = 0.144
samples = 413
value = [602, 51]
class = 1
gini = 0.367
samples = 65
value = [24, 75]
class = 2
gini = 0.004
samples = 4133
value = [6576, 14]
class = 1
gini = 0.444
samples = 1498
value = [784, 1572]
class = 2
gini = 0.011
samples = 3645
value = [32, 5728]
class = 2
gini = 0.438
samples = 571
value = [601, 288]
class = 1
gini = 0.497
samples = 1096
value = [928, 795]
class = 1
gini = 0.06
samples = 5733
value = [285, 8858]
class = 2
gini = 0.5
samples = 1383
value = [1074, 1084]
class = 2
gini = 0.467
samples = 217
value = [126, 214]
class = 2

[28]
```
pipe = make_pipeline(
    TargetEncoder(),
    RandomForestClassifier(random_state=2,n_jobs=-1,max_depth=4, n_estimators=250, min_samples_split=10)
)
```

```
permuter = PermutationImportance (
    pipe.named_steps['randomforestclassifier'],
    scoring = 'f1',
    n_iter=5,
    random_state=2
)
```

[30]
```
pipe.fit(X_train,y_train)
```

```
Pipeline(steps=[('targetencoder',
                 TargetEncoder(cols=['firstBlood', 'firstTower',
                                     'firstInhibitor', 'firstBaron',
                                     'firstDragon', 'firstRiftHerald',
                                     'gametime'])),
                ('randomforestclassifier',
                 RandomForestClassifier(max_depth=4, min_samples_split=10,
                                        n_estimators=250, n_jobs=-1,
                                        random_state=2))])
```

```
y_pred = pipe.predict(X_train)

print(classification_report(y_train,y_pred))
```

```
              precision    recall  f1-score   support

           1       0.97      0.98      0.97     20435
           2       0.97      0.97      0.97     19774

    accuracy                           0.97     40209
   macro avg       0.97      0.97      0.97     40209
weighted avg       0.97      0.97      0.97     40209
```

firstBlood	firstTower	firstInhibitor	firstBaron	firstDragon	firstRiftHerald	t1_towerKills	t1_inhibitorKills
Blue	Red	Red	Red	Red	Red	2	2

t1_baronKills	t1_dragonKills	t2_towerKills	t2_inhibitorKills	t2_baronKills	t2_dragonKills
1	4	7	2	1	4

```
y_pred = pipe.predict_proba(scen.iloc[[1]])

print(f'Blue가 이길 확률 : {round(y_pred[0][0]*100,2)}%')
print(f'Red가 이길 확률 : {round(y_pred[0][1]*100,2)}%')
```

```
Blue가 이길 확률 : 31.77%
Red가 이길 확률 : 68.23%
```

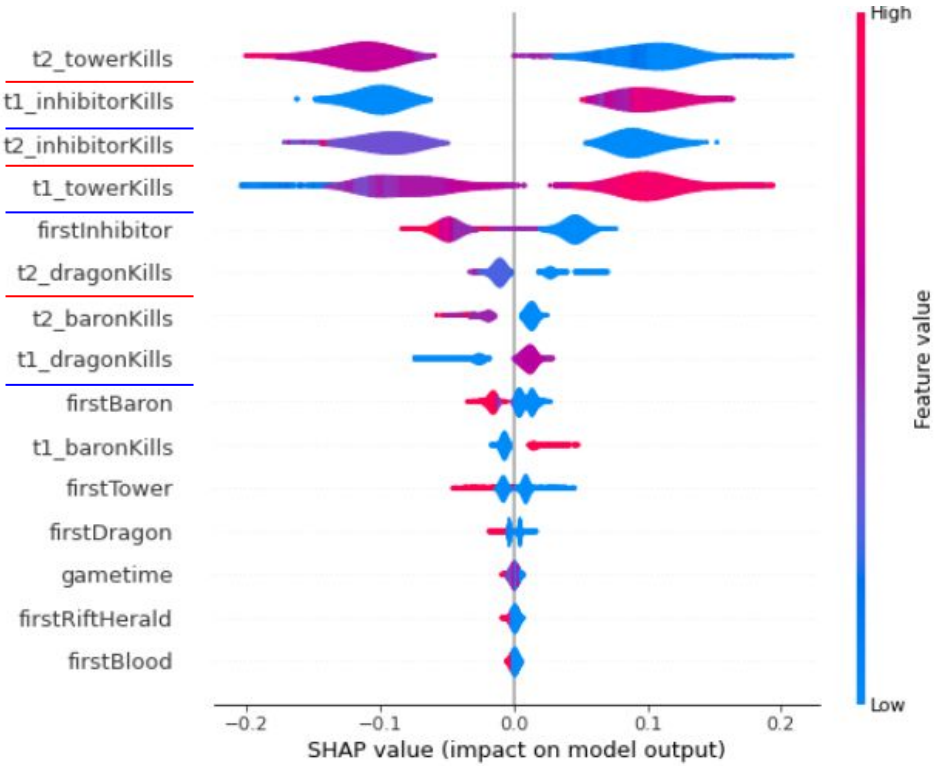
firstBlood	firstTower	firstInhibitor	firstBaron	firstDragon	firstRiftHerald	t1_towerKills	t1_inhibitorKills
None	None	None	None	None	None	0	0

t1_baronKills	t1_dragonKills	t2_towerKills	t2_inhibitorKills	t2_baronKills	t2_dragonKills
0	0	0	0	0	0

```python
y_pred = pipe.predict_proba(scen.iloc[[0]])

print(f'Blue가 이길 확률 : {round(y_pred[0][0]*100,2)}%')
print(f'Red가 이길 확률 : {round(y_pred[0][1]*100,2)}%')
```

```
Blue가 이길 확률 : 65.28%
Red가 이길 확률 : 34.72%
```


t2_towerKills
t1_inhibitorKills
t2_inhibitorKills
t1_towerKills
firstInhibitor
t2_dragonKills
t2_baronKills
t1_dragonKills
firstBaron
t1_baronKills
firstTower
firstDragon
gametime
firstRiftHerald
firstBlood
−0.2
−0.1
0.0
0.1
0.2
SHAP value (impact on model output)
High
Feature value
Low

END