Game Result Prediction

탁성원

최종작성일 : 21.10.24

Contents

주제 선정 배경

주제 선정 배경

전 세계에서 가장 유명하다고 할 수 있는 게임 League of Legends
다양한 요소가 존재하는 게임으로 데이터에 대한 흥미가 생겨 분석 및 예측 시도

게임 목표

Red 진영과 Blue 진영의 5 : 5 팀 배틀
상대방의 적 본진인 **넥서스(Nexus) 격파 목표**

맵 내 미니언, 타워, 억제기 등의 다양한 요소 존재

대표적 전장, <소환사의 협곡>

분석 목표 및 문제 정의

분석 목표

Kaggle 출처의 League of Legends Game Data 활용
인게임 데이터를 통한 승리 팀 예측 모델 생성

문제 정의

최상위 티어, 챌린저 티어 마크

분석 대상 / 챌린저 랭크 간의 게임 총 26,904건
분석 목표 / Blue 진영과 Red 진영 중 승리 진영의 예측
평가 지표 / 정확도(Accuracy) 기반 평가 실시

승패 판단의 상호 대칭적 문제이므로 해당 지표 1개로 충분

사전 데이터 탐색 I

데이터 탐색

26,904 건의 경기에 대한 In-game 데이터로 Blue 진영과 Red 진영 각각의 데이터 존재

★ Blue 진영의 결과에 따른 Red 진영의 대칭적인 결과 데이터 포함

ex) BlueWins가 1(승리) 일 경우, 자연스럽게 RedWins는 0(패배)일 것이다.

기본 정보 파악

```
df.blueWins.value_counts()

1    13454
0    13450
```

Blue 진영과 Red 진영의 승률은 거의 동일하다.

```
df.gameDuraton.describe()

count    26904.000000
mean      1448.653657
std        422.577288
min        190.000000
25%       1152.000000
50%       1435.000000
75%       1738.000000
max       3301.000000
```

게임은 평균 약 1440초, 약 24분 정도 진행된다.

```
sum(df.isna().sum())

0
```

데이터 내 결측치는 별도로 존재하진 않았음.

사전 데이터 탐색 II

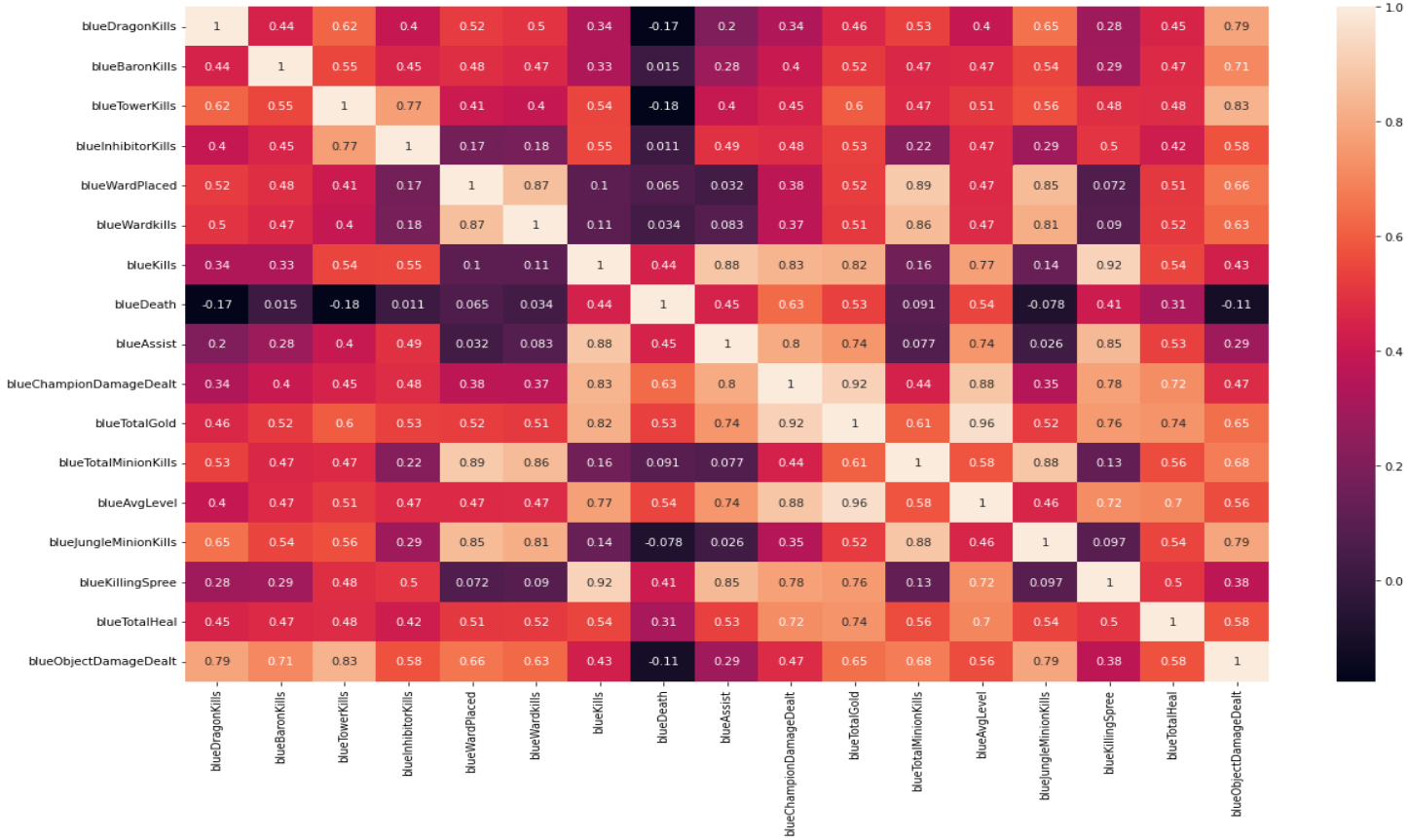
변수 선택

Blue 진영의 게임 데이터를 바탕으로 한 게임 내 변수 간 상관관계 파악

1] 우리 팀 킬수 = 상대팀 데스 수
-> 데스 수 제외
2] 어시스트는 킬에 직접적 영향
-> 어시스트 수 제외
3] 골드는 다른 킬수 등에 비례적 관계
-> 누적 골드 제외
4] 레벨 또한 미니언 사냥 및 적 챔피언 킬 등의 결과
-> 레벨 제외

변수 선택 결과

```
df_X2 = df[['gameDuraton', 'blueFirstBlood', 'blueFirstTower',
            'blueFirstBaron', 'blueFirstDragon', 'blueFirstInhibitor',
            'blueDragonKills', 'blueBaronKills', 'blueTowerKills',
            'blueInhibitorKills', 'blueWardPlaced', 'blueWardkills',
            'blueKills', 'blueDeath', 'blueChampionDamageDealt',
            'blueTotalMinionKills',
            'blueJungleMinionKills', 'blueKillingSpree', 'blueTotalHeal',
            'blueObjectDamageDealt',
            'redDragonKills', 'redBaronKills', 'redTowerKills',
            'redInhibitorKills', 'redWardPlaced', 'redWardkills',
            'redChampionDamageDealt',
            'redTotalMinionKills',
            'redJungleMinionKills', 'redKillingSpree', 'redTotalHeal',
            'redObjectDamageDealt']]
```



Blue팀 기준 인게임 내 지표간 상관관계

사전 데이터 탐색 III

선취점 데이터 분석

데이터 내 선취점 관련 변수들 중 각각을
단독적으로 활용했을 때의 예측 정확도 확인

```
first_list = ['blueFirstBlood', 'blueFirstTower',
              'blueFirstBaron', 'blueFirstDragon', 'blueFirstInhibitor']
for i in range(5):
  count = 0
  for j in range(26903):
    if df.loc[j,'blueWins'] == df_X2.loc[j, first_list[i]]:
      count += 1
  print(count / 26903 * 100)
```

결과

FirstBlood
60.32%

FirstTower
72.42%

FirstBaron
65.75%

FirstDragon
63.15%

FirstInhibitor
80.73%

타 선취점 데이터 대비 FirstTower, FirstInhibitor의 예측 성능이 매우 우수
선취점 데이터 중 해당 두 데이터만 우선적으로 반영 시도.

분석 아이디어

2 분석 요약

기본 아이디어

관찰

아무튼 Blue팀과 Red팀 중 **승리팀을** 찾으면 되는 것이다.

캐릭터들이 입힌 피해의 총량 등 각 팀별 인게임내 변수 별로
Blue팀과 Red팀 중 **더 우세한 팀이 존재할 것이다.**

게임은 블루와 레드의 힘겨루기 양상.

방법

Blue팀과 Red팀에 공통적으로 존재하는 변수 대상

Blue팀이 Red팀에 비해 특정 변수에 대해
더 높은 값을 가질 경우 1
아닐 경우 0

즉, 최초 킬 여부, 최초 타워 격파 여부 처럼 모든 인게임 요소가 0-1 변수로 나오게 된다.

변환된 변수를 바탕으로 분류 문제의 주요 모델 적용

데이터 전처리

2 분석 요약

변수 정리 과정

```
df_X2 = df[['gameDuraton', 'blueFirstBlood', 'blueFirstTower',
'blueFirstBaron', 'blueFirstDragon', 'blueFirstInhibitor',
'blueDragonKills', 'blueBaronKills', 'blueTowerKills',
'blueInhibitorKills', 'blueWardPlaced', 'blueWardkills',
'blueKills', 'blueDeath', 'blueChampionDamageDealt',
'blueTotalMinionKills',
'blueJungleMinionKills', 'blueKillingSpree', 'blueTotalHeal',
'blueObjectDamageDealt',
'redDragonKills', 'redBaronKills', 'redTowerKills',
'redInhibitorKills', 'redWardPlaced', 'redWardkills',
'redChampionDamageDealt',
'redTotalMinionKills',
'redJungleMinionKills', 'redKillingSpree', 'redTotalHeal',
'redObjectDamageDealt']]
```

선취/시간 데이터

Blue 데이터

Red 데이터

FirstTower, First Inhibitor만 선택,
Blue가 1이면 Red가 0이므로 Blue에 대해서만 추출

Blue가 클 경우 1, 작을 경우 0

```
df_X2['dargonkill_cmprsn'] = [1 if df_X2['blueDragonKills'][i] >= df_X2['redDragonKills'][i] else 0 for i in range(len(df_X2['blueKills']))]
df_X2['baronkill_cmprsn'] = [1 if df_X2['blueBaronKills'][i] >= df_X2['redBaronKills'][i] else 0 for i in range(len(df_X2['blueKills']))]
df_X2['towerkill_cmprsn'] = [1 if df_X2['blueTowerKills'][i] >= df_X2['redTowerKills'][i] else 0 for i in range(len(df_X2['blueKills']))]
df_X2['inhibitorkill_cmprsn'] = [1 if df_X2['blueInhibitorKills'][i] >= df_X2['redInhibitorKills'][i] else 0 for i in range(len(df_X2['blueKills']))]
df_X2['wardplaced_cmprsn'] = [1 if df_X2['blueWardPlaced'][i] >= df_X2['redWardPlaced'][i] else 0 for i in range(len(df_X2['blueKills']))]
df_X2['wardkill_cmprsn'] = [1 if df_X2['blueWardkills'][i] >= df_X2['redWardkills'][i] else 0 for i in range(len(df_X2['blueKills']))]
df_X2['champdeal_cmprsn'] = [1 if df_X2['blueChampionDamageDealt'][i] >= df_X2['redChampionDamageDealt'][i] else 0 for i in range(len(df_X2['blueKills']))]
df_X2['minionkill_cmprsn'] = [1 if df_X2['blueTotalMinionKills'][i] >= df_X2['redTotalMinionKills'][i] else 0 for i in range(len(df_X2['blueKills']))]
df_X2['jungminikill_cmprsn'] = [1 if df_X2['blueJungleMinionKills'][i] >= df_X2['redJungleMinionKills'][i] else 0 for i in range(len(df_X2['blueKills']))]
df_X2['spree_cmprsn'] = [1 if df_X2['blueKillingSpree'][i] >= df_X2['redKillingSpree'][i] else 0 for i in range(len(df_X2['blueKills']))]
df_X2['heal_cmprsn'] = [1 if df_X2['blueTotalHeal'][i] >= df_X2['redTotalHeal'][i] else 0 for i in range(len(df_X2['blueKills']))]
df_X2['objdeal_cmprsn'] = [1 if df_X2['blueObjectDamageDealt'][i] >= df_X2['redObjectDamageDealt'][i] else 0 for i in range(len(df_X2['blueKills']))]
```

결과

	kill_ Cmprsn	Dragon kill_ cmprsn	baronkil_ cmprsn	towerkil_ cmprsn	Inhibitor kill_ cmprsn	Ward Placed _cmprsn	wardkill_ cmprsn	Champ deal_ cmprsn	Minion kill_ Cmprsn	jungmini kill_ cmprsn	spree_ cmprsn	heal_ cmprsn	objdeal_ cmprsn	blueFirst Tower	blueFirst Inhibitor
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0
4	0	0	1	1	1	1	0	1	1	1	0	0	1	1	1

세부 분석 과정 I

데이터셋 분리

전체 데이터 셋을 Train Set, Test Set으로 나눠 학습, 검증을 별도로 진행

적용 모델

Decision Tree

Logistic
Regression

LightGBM

세부 분석 과정 II

결정트리 적용 가장 기본적인 분류 모델인 Decision Tree 사용

근거 빠른 계산 속도 및 간단한 모델을 바탕으로 결과에 대한 해석 용이
시각화를 통한 분류 과정 확인 가능

모델 최대 깊이 4인 결정 트리 모델 생성

결과

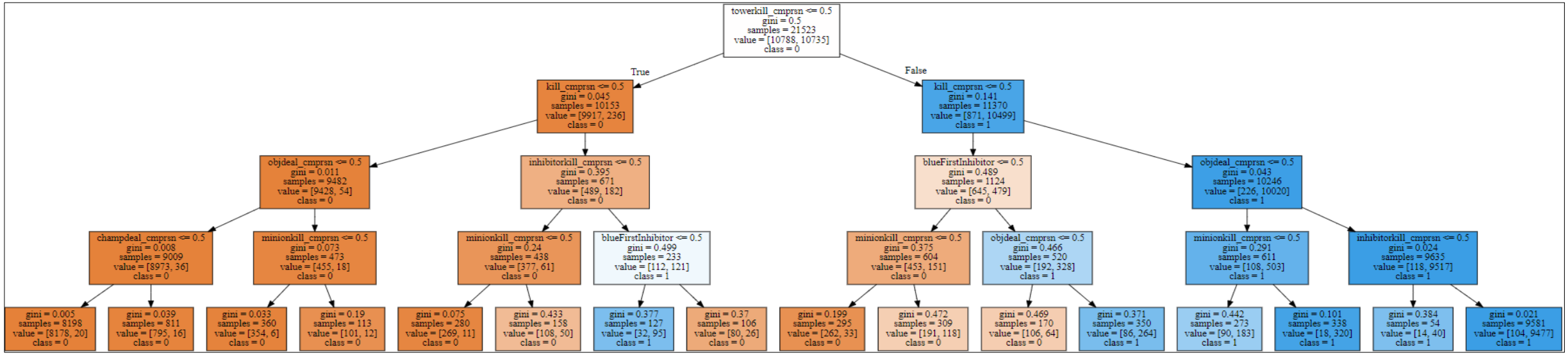
```
# 결정트리 학습
decision_tree = DecisionTreeClassifier(random_state = 0,
                                       max_depth=4,
                                       criterion='gini')
decision_tree = decision_tree.fit(X_train,y_train)
pred = decision_tree.predict(X_test)
accuracy_score(pred, y_test)
```

Result : 97.06%

세부 분석 과정 III

2 분석 요약

결정트리 시각화



분석 분류에 다음 7개의 변수만 사용되었음을 확인

1. towerkill_cmprsn
2. kill_cmprsn
3. champdeal_cmprsn
4. blueFirstInhibitor
5. inhibitorkill_cmprsn
6. minionkill_cmprsn
7. objdeal_cmprsn

추가시도 :
7개의 변수에 대한 다수결 실시

```
count = 0
for i in range(26903):
  if int(df_X2.loc[i,'kill_cmprsn'])+ int(df_X2.loc[i,'towerkill_cmprsn'])+\
  int(df_X2.loc[i,'inhibitorkill_cmprsn'])+ int(df_X2.loc[i,'champdeal_cmprsn'])+\
  int(df_X2.loc[i,'minionkill_cmprsn'])+ int(df_X2.loc[i,'objdeal_cmprsn'])+\
  int(df_X2.loc[i,'blueFirstInhibitor']) >= 4:
    if df.loc[i,'blueWins'] == 1:
      count += 1
  else:
    if df.loc[i, 'blueWins'] == 0:
      count += 1

print(count/26903 * 100)
```

Result : 96.73%

세부 분석 과정 IV

2 분석 요약

로지스틱 회귀

모델 Cut_off = 0.5 기준의 기본적인 로지스틱 회귀 모델 적용

(Blue와 Red의 대칭적 상황이므로 Cut_off의 조절 필요 없음)

```
# 로지스틱 회귀
model = LogisticRegression()
model.fit(X_train, y_train)
pred = model.predict(X_test)
accuracy_score(pred, y_test)
```

Result : 97.64%

해석 모델의 계수를 출력해보면 다음과 같다.

```
# model.coef_ 2차원이므로 1차원으로 만든다.
coef_list = np.concatenate(model.coef_).tolist()
for i in range(len(coef_list)):
  print(X_train.columns[i] + ' 계수 : ' + str(round(coef_list[i], 3)))
```

```
kill_cmprsn 계수 : 2.365
dargonkill_cmprsn 계수 : -0.242
baronkill_cmprsn 계수 : 0.61
towerkill_cmprsn 계수 : 2.786
inhibitorkill_cmprsn 계수 : 1.148
wardplaced_cmprsn 계수 : 0.405
wardkill_cmprsn 계수 : -0.227
champdeal_cmprsn 계수 : 1.687
minionkill_cmprsn 계수 : 2.034
jungminikill_cmprsn 계수 : 0.061
spree_cmprsn 계수 : 0.552
heal_cmprsn 계수 : 1.427
objdeal_cmprsn 계수 : 1.63
blueFirstTower 계수 : -0.533
blueFirstInhibitor 계수 : 1.094
```

앞에서 결정트리에서 살펴 본 7개의 변수

1. towerkill_cmprsn
2. kill_cmprsn
3. champdeal_cmprsn
4. blueFirstInhibitor
5. inhibitorkill_cmprsn
6. minionkill_cmprsn
7. objdeal_cmprsn

중에서도 *kill, tokwerkill, minionkill에 대한 변수의 계수가 크게 나타났다.*

세부 분석 과정 V

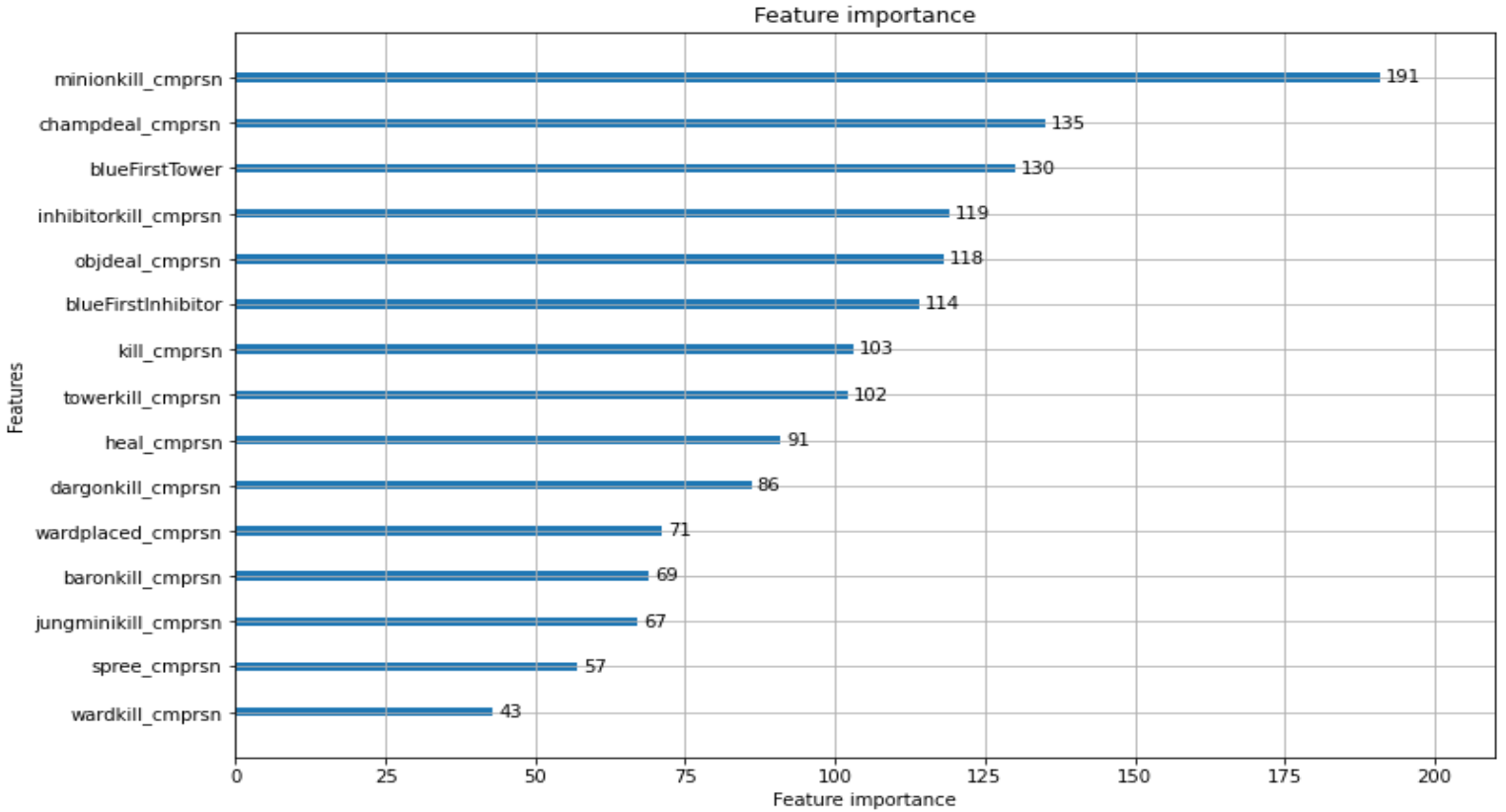
2 분석 요약

LightGBM

모델 앙상블 기법 중 하나인 LightGBM을 활용

```
lgbm = LGBMClassifier(n_estimators = 100, max_depth = 4, application = 'binary')
lgbm.fit(X_train, y_train)
lgbm_pred = lgbm.predict(X_test)
accuracy_score(lgbm_pred, y_test)
```

Result : 97.97%



결정트리에서 본 7개의 변수
1. towerkill_cmprsn
2. kill_cmprsn
3. champdeal_cmprsn
4. blueFirstInhibitor
5. inhibitorkill_cmprsn
6. minionkill_cmprsn
7. objdeal_cmprsn 이외에
'blueFirstTower'도 큰 영향을 준 변수.

나머지 7개의 변수 모두
Feature Importance 상위 8개에 해당

모델 개선 시도

모델 개선 과정 : HyperParameter Tuning

방법

위의 방법들 중 성능이 가장 좋았던 LightGBM 활용
HyperParameter를 조절. 최적의 정확도를 보이는 HyperParameter 선택

```
from sklearn.model_selection import GridSearchCV
model_lgbm_cv = LGBMClassifier(application = 'binary')
params = {'max_depth':[3, 4, 5, 6],
          'learning_rate':[0.01, 0.05, 0.07, 0.1, 0.2],
          'num_iters' : [50, 100, 200]}
grid = GridSearchCV(estimator=model_lgbm_cv,
                    param_grid=params,
                    scoring='accuracy',
                    cv=5)
grid.fit(X_train, y_train)
print("최적 파라미터: ", grid.best_params_)
```

learning_rate = 0.05, max_depth = 5
num_iters = 500이 최적으로 선정

```
model_lgbm_cv = LGBMClassifier(application = 'binary',
                               max_depth = 5,
                               learning_rate = 0.05,
                               num_iters = 50)
model_lgbm_cv.fit(X_train, y_train)
pred = model_lgbm_cv.predict(X_test)
accuracy_score(pred, y_test)
```

Result : 97.88%
오히려 CV를 고려하지 않은 상황보다 나쁘다?
약간의 오버피팅이 예상된다.

그랜드마스터 랭크에의 적용

3 결과 적용

그랜드마스터 랭크에는?

그랜드마스터 랭크 경기 65,896건에 대해
앞에서 만든 모델을 동일하게 적용할 수 있을지 검증

모델 사용

Decision Tree 모델을 동일하게 적용
Accuracy : 96.52%

```
res_pred = decision_tree.predict(gm_df_game_factor)
accuracy_score(gm_df_y, res_pred)
```

```
0.9652179191453198
```

추가 검증

Decision Tree 생성시, 일부 변수로 다수결을 활용한 예측 실시
동일하게 적용가능한지 확인

```
count = 0
for i in range(65896):
  if int(gm_df.loc[i,'kill_cmprsn'])+₩
  int(gm_df.loc[i,'towerkill_cmprsn'])+₩
  int(gm_df.loc[i,'inhibitorkill_cmprsn'])+₩
  int(gm_df.loc[i,'champdeal_cmprsn'])+₩
  int(gm_df.loc[i,'minionkill_cmprsn'])+₩
  int(gm_df.loc[i,'objdeal_cmprsn'])+₩
  int(gm_df.loc[i,'blueFirstInhibitor']) >= 4:
    if gm_df.loc[i,'blueWins'] == 1:
      count += 1
  else:
    if gm_df.loc[i, 'blueWins'] == 0:
      count += 1

print(count/65896 * 100)
```

Result : 96.44%

더 하위의 랭크에 대해서는 추가 검증해야 하나,
어느 정도 승부 예측에 유의미한 지표로 사용하기엔
유의미하며, ML기법을 사용하지 않고 예측할 수 있어
예측을 매우 빠른 시간내에 할 수 있다.

결론 및 분석 개선 방향

결론

승패의 예측을 위해 다음 7가지 변수를 비교, 더 많은 변수가 우수한 팀이 승리할 가능성이 높다.

분석 발전 방향

[1] 변수별 blue/red 우세 여부가 아닌 어느 정도의 비율로 우세인지를 계산해볼 수 있을 것이다.
[2] 성장 정도가 유사한 경우에 승패 예측이 어려운데, 이 경우를 구분할 지표를 추가 생성해본다.