OSSP Progress Report 3

<리그오브레전드 승/패 상관관계분석>

DHL

2016112190 한종호

2016112204 임정우

2016112231 정현성

2016112194 최재원

1. 프로젝트 개요

- 라이엇 게임즈 API를 통해 데이터수집 (완료)
- **EDA 진행(진행중)**
- EDA 된 데이터기반 모델링 진행
- 모델링 분석 결과 시각화
- **프로그램 로직 및 UI 개발(진행중)**
- 버그 수정 및 유닛 테스트

2. 3주차 프로젝트 진행상황

1) 미해결 이슈

1-1) 라이엇API에서 받아온 데이터들의 컬럼

	gameId	platformId	gameCreation	gameDuration	queueId	mapId	seasonId	gameVersion	gameMode	gameType	teams	participants
0	4771375338	KR	1604844366180	1075	420	11	13	10.22.341.643	CLASSIC	MATCHED_GAME	[{'teamId': 100, 'win': 'Win', 'firstBlood': T...	[{'participantId': 1, 'teamId': 100, 'champion...
1	4770146326	KR	1604818960847	1606	420	11	13	10.22.341.643	CLASSIC	MATCHED_GAME	[{'teamId': 100, 'win': 'Fail', 'firstBlood': ...	[{'participantId': 1, 'teamId': 100, 'champion...
2	4767376134	KR	1604733123324	1069	420	11	13	10.22.341.643	CLASSIC	MATCHED_GAME	[{'teamId': 100, 'win': 'Win', 'firstBlood': T...	[{'participantId': 1, 'teamId': 100, 'champion...
3	4767361959	KR	1604729191092	1890	420	11	13	10.22.341.643	CLASSIC	MATCHED_GAME	[{'teamId': 100, 'win': 'Win', 'firstBlood': F...	[{'participantId': 1, 'teamId': 100, 'champion...
4	4761504494	KR	1604485121523	925	420	11	13	10.22.341.643	CLASSIC	MATCHED_GAME	[{'teamId': 100, 'win': 'Fail', 'firstBlood': ...	[{'participantId': 1, 'teamId': 100, 'champion...

gameid : 게임을 식별하기 위한 ID 값

platformId : 해당 게임이 어느 나라 채널인지를 알 수 있는 값

gameCreation : 챔피언 선택이 모두 마치고, 게임이 시작한 시점이 아닌 리그오브레전드 프로그램이 실행시켜지는 시점의 시간 값

gameDuration : 총 게임에 걸린 시간(초)

queueID : 게임의 유형(420 - 솔로랭크, 450 - 팀랭크 등등)

mapID : 게임의 장소에 해당 하는 값

seasonID : 게임 내에 존재하는 시즌을 나타내는 값

gameversion : 게임버전

gameMode : 게임모드

gameType : 게임유형 (랜덤으로 만난 건지, 직접 초대해서 만난건지)

teams : 팀 별 해당 하는 정보(승패 여부, 첫 킬, 첫 타워파괴, 총 드래곤 킬수, 얻은 총 킬 포인트 등등)

participants : 10명의 유저에 관한 게임정보(챔피언 정보, 스킬 정보, 능력치 정보, 게임내에서 킬한 횟수, 획득한 골드 정보, 상대에게 입힌 데미지 정보, 구입한 아이템정보 등등)

participantsIdentities : 10명의 유저에 관한 아이디정보(소환사이름, 소환사ID, accountID, PlatformID

1-2) 게임에 영향을 주는 사항(점수)

게임 내 킬스코어가 존재하긴 하되, 이것은 다른 캐주얼게임처럼 일정점수에 도달했을 때 승패가 나눠지는 것이 아닌 게임의 척도를 나타내는 것이기 때문에 무조건적으로 게임자체에 영향을 준다고 설명할 순 없다. 라인전에서 챔피언 간

상성으로 인하여 킬이 나서 스코어가 오르게 되고 -> 점점 격차가 벌어지면서 최종적으로 상대편 진영을 먼저 모두 파괴하는 것이 게임이 끝나는 흐름으로 봤을 때 킬 스코어는 게임의 현 상황에서 어느 팀이 더 유리한지 판단은 가능하나 그 반대로 킬 스코어가 오르는 것이 게임에 영향을 주지는 않는다.

2) EDA 과정에서 이상치 판별

- **기존의 라인별 챔피언 리스트 -> 딕셔너리로 변경**

```
for k, i in carry_dict.items():
    print(f'key : {k:<3} | id : {i}')
```

```
key : 523 | id : Aphelios
key : 22  | id : Ashe
key : 51  | id : Caitlyn
key : 119 | id : Draven
key : 81  | id : Ezreal
key : 202 | id : Jhin
key : 222 | id : Jinx
key : 145 | id : Kaisa
key : 429 | id : Kalista
key : 96  | id : KogMaw
key : 236 | id : Lucian
key : 21  | id : MissFortune
key : 360 | id : Samira
key : 15  | id : Sivir
key : 18  | id : Tristana
```

< carry_dict (원거리 딜러 딕셔너리 예시) >

champion의 **key(int)**값과 **id(string)**를 둘다 갖기위해서
추후에 **id를 쓸 상황을 대비** -> 아직 계획엔 없음

- **이상치(Outlier)**

- 이상치 역시 분석 결과를 왜곡시키기 때문에 **이상치를 대체하거나, 제거**해야 좀 더 정확한 결과를 가져올 수 있다. 보통은 포지션 별로 같은포지션 챔피언끼리 상대를 하는데 다른 포지션의 챔피언을 상대하는 경우도 있다.

  예를 들어 Mid 포지션의 챔피언이 원거리 딜러에 등장 하는 경우가 있다. 이런 경우는 원거리 딜러가 Mid 챔피언을 상대하는데, 그런 경우가 적어서 경기 수 가 충분하지 못하다. 이러한 경우에는 적은 경기의 수를 갖고 승률을 판단하기에는 어려움이 있다.

그래서 이상치(적은 판수)들의 기준을 정하고, 승리와 패배를 50: 50비율로 정하여 대체할 계획이다.

- **기존의 승률을 구하는 win_rate 함수의 기준 경기값 (25 경기)이하 수정중**

```
def win_rate(line_df):
    for champ in line_df:
        for 상대 in line_df[champ][:].index:
            if (numberWinsChampion[상대][champ] + numberWinsChampion[champ][상대]) < 25: 승률 = 0.5
```

기준값=?

  - 기존에 임의로 25경기로 정한값을, 각 라인별 경기 평균 횟수(mean), 표준편차(std), 4사분위값등을 이용해서 기준 경기값 수정예정...

Top

```
1 top_games.describe()
```

```
count    780.000000
mean      22.116667
std       43.692712
min        0.000000
25%        2.000000
50%        7.000000
75%       23.000000
max      476.000000
```

<Top 챔피언들의 경기횟수에 대한 describe()>

- 포지션별로 win_rate 기준경기를 다르게 적용후 포지션별로 승률 도출
- 포지션 모두 같은 기준경기 값을 갖도록 적용후 전체에 대해서 승률 도출
- 위 두가지의 결과로 모델링 진행 예정

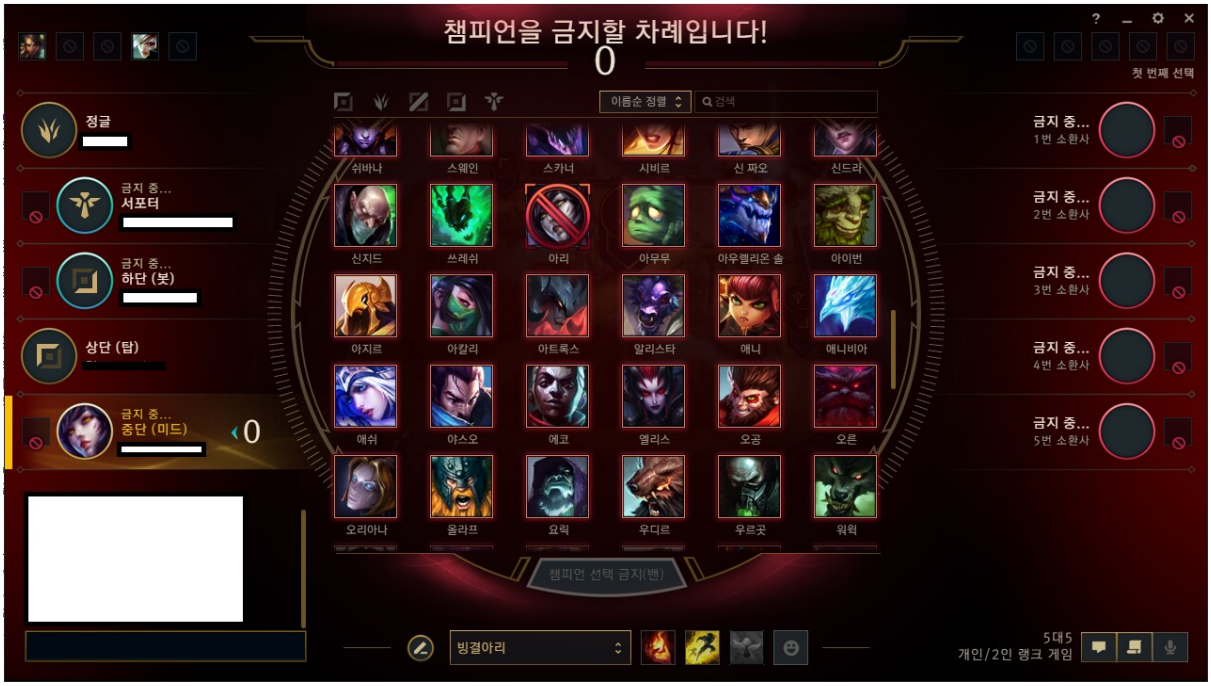
3) GUI

- tkinter 모듈을 사용해서 GUI 제작 예정

tkinter 패키지(《Tk 인터페이스》)는 Tk GUI 툴킷에 대한 표준 파이썬 인터페이스입니다. Tk와 tkinter는 대부분의 유닉스 플랫폼과 윈도우 시스템에서 사용할 수 있습니다. (Tk 자체는 파이썬 일부가 아닙니다; ActiveState에서 유지 보수됩니다.)

(https://docs.python.org/ko/3/library/tkinter.html)

- exe 파일 형태로 제작 예정이기 때문에, 그에 맞는 파이썬 GUI 툴들을 찾아봄. (pyqt, tkinter, pyside, pygtk 등)-> pyqt와 tkinter가 가장 많이 사용되는 GUI 툴이고, 별도의 설치없이 사용할 수 있는 tkinter가 제작하기 편하다고 판단되어 tkinter를 선택.
- 실제 게임 내부에서 나타나는 밴픽 화면과 가장 비슷하게 구현.
- 추가적으로 프로젝트의 주된 기능인 챔피언 추천을 보여줄 수 있는 화면 구현.

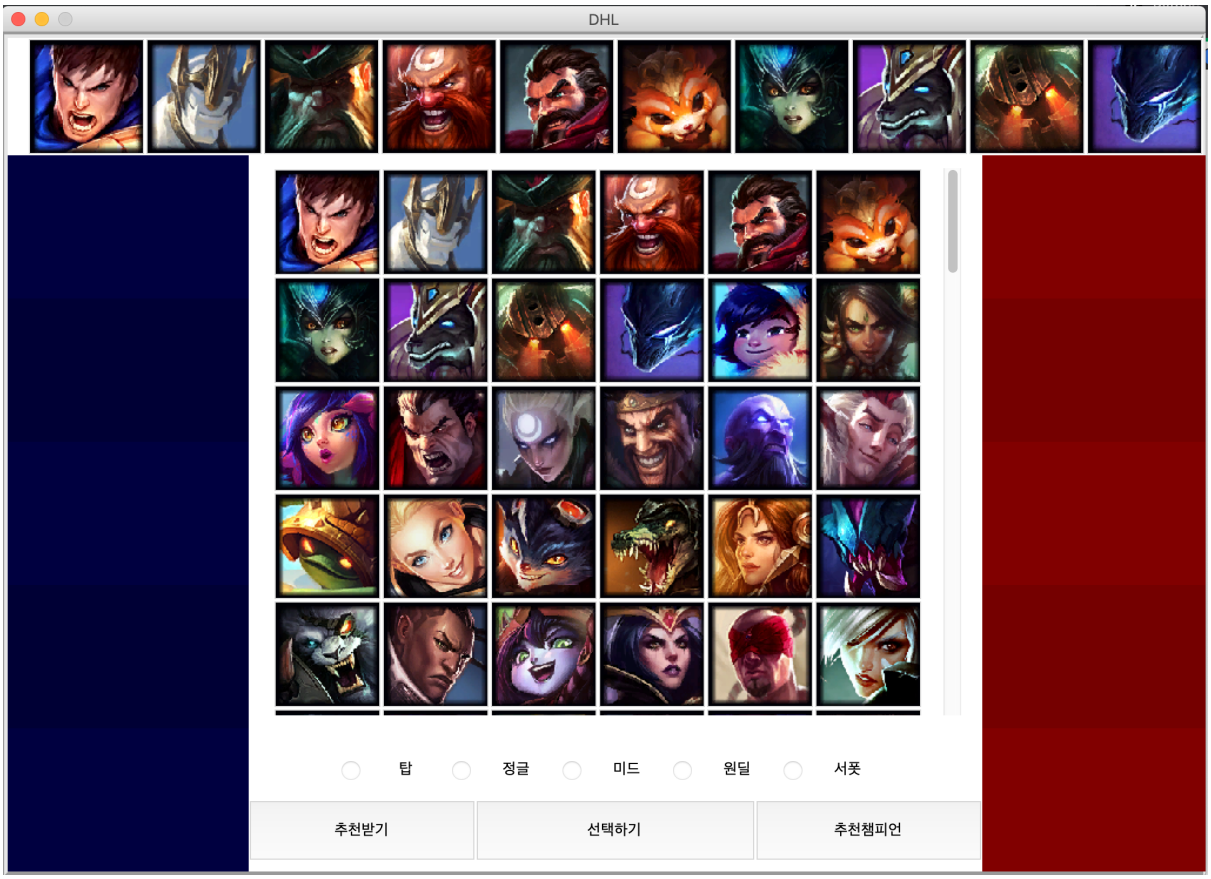


- 실제 Pick & Ban 화면

왼쪽에 플레이어의 팀, 오른쪽에 상대팀이 있고, 순서대로 픽을 진행함.

아래쪽에는 채팅창과 게임에 필요한 다른 인터페이스가 있으나 구현하고자 하는 프로젝트에서 사용되지 않을 부분이므로, 이 부분에 챔피언 추천 인터페이스 구현.

- 현재 진행된 GUI



상단에는 밴당한 챔피언 10개가 출력되고, 아래에서 각 팀에서 선택한 챔피언을 양 옆에 입력할 수 있도록 함. 최하단에는 버튼을 나열해 원하는 작업을 수행할 수 있게 함.