OSSP Progress Report 2

<리그오브레전드 승/패 상관관계분석>

DHL

2016112190 한종호

2016112204 임정우

2016112231 정현성

2016112194 최재원

1. 프로젝트 개요

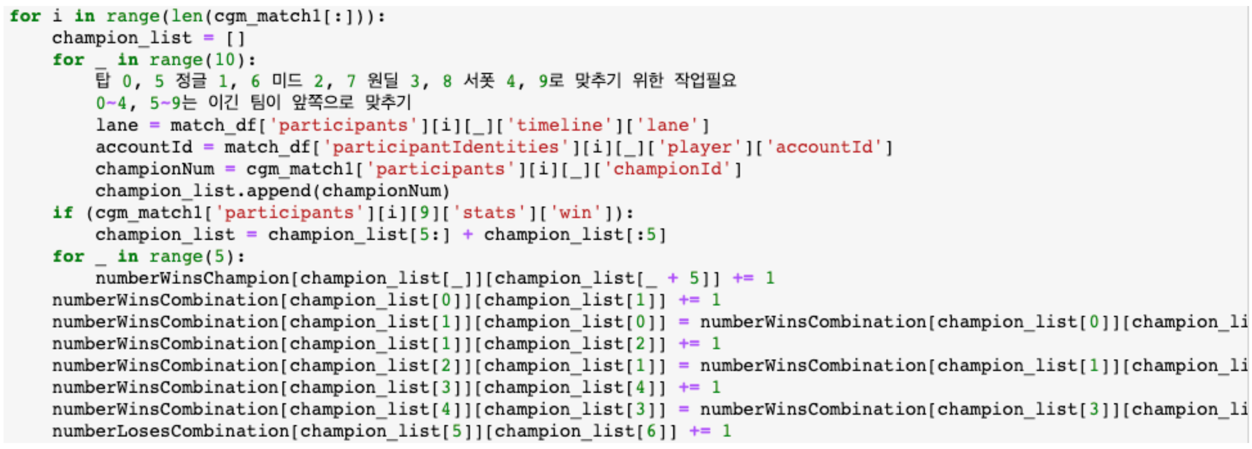
- 라이엇 게임즈 API를 통해 데이터수집
- **EDA 진행 (진행중)**
- EDA 된 데이터기반 모델링 진행
- 모델링 분석 결과 시각화
- 프로그램 로직 및 UI 개발
- 버그 수정 및 유닛 테스트

2. 2주차 프로젝트 진행상황

- Riot API를 통해서 약 90,000개의 데이터 수집완료
- 각 유저의 최근 100게임의 전적을 가지고 챔피언에 따른 승률 분석

```
[523, 89, 84, 54, 120, 91, 164, 517, 80, 202]
[89, 79, 523, 164, 91, 412, 246, 202, 76, 85]
[164, 81, 43, 4, 76, 114, 3, 875, 421, 22]
[412, 523, 875, 876, 41, 98, 53, 142, 266, 22]
[875, 235, 517, 35, 236, 85, 141, 81, 39, 44]
[142, 104, 89, 58, 81, 876, 523, 80, 133, 517]
[202, 59, 3, 89, 24, 875, 523, 5, 84, 517]
[74, 89, 360, 76, 246, 517, 64, 142, 80, 235]
[64, 523, 4, 164, 44, 421, 24, 142, 202, 89]
[236, 202, 80, 245, 24, 517, 523, 875, 246, 27]
[21, 141, 245, 80, 58, 29, 497, 104, 74, 61]
[64, 117, 523, 24, 61, 412, 81, 76, 157, 164]
[24, 203, 4, 111, 360, 202, 12, 104, 142, 74]
[78, 89, 141, 523, 3, 117, 80, 91, 876, 202]
[141, 103, 80, 22, 164, 74, 412, 876, 58, 222]
[61, 516, 360, 875, 120, 89, 58, 163, 103, 81]
[64, 25, 202, 84, 517, 523, 203, 89, 91, 74]
[91, 76, 18, 106, 80, 516, 12, 202, 64, 246]
[58, 523, 246, 412, 245, 117, 236, 76, 22, 53]
[876, 67, 91, 432, 68, 58, 76, 246, 81, 111]
```

<한유저의 최근 20경기에 등장했던 챔피언 고유id 출력 (한 게임당 10개의 챔피언 사용)>



```
for i in range(len(cgm_match1[:])):
    champion_list = []
    for _ in range(10):
        탑 0, 5 정글 1, 6 미드 2, 7 원딜 3, 8 서폿 4, 9로 맞추기 위한 작업필요
        0~4, 5~9는 이긴 팀이 앞쪽으로 맞추기
        lane = match_df['participants'][i][_]['timeline']['lane']
        accountId = match_df['participantIdentities'][i][_]['player']['accountId']
        championNum = cgm_match1['participants'][i][_]['championId']
        champion_list.append(championNum)
    if (cgm_match1['participants'][i][9]['stats']['win']):
        champion_list = champion_list[5:] + champion_list[:5]
    for _ in range(5):
        numberWinsChampion[champion_list[_]][champion_list[_ + 5]] += 1
    numberWinsCombination[champion_list[0]][champion_list[1]] += 1
    numberWinsCombination[champion_list[1]][champion_list[0]] = numberWinsCombination[champion_list[0]][champion_li
    numberWinsCombination[champion_list[1]][champion_list[2]] += 1
    numberWinsCombination[champion_list[2]][champion_list[1]] = numberWinsCombination[champion_list[1]][champion_li
    numberWinsCombination[champion_list[3]][champion_list[4]] += 1
    numberWinsCombination[champion_list[4]][champion_list[3]] = numberWinsCombination[champion_list[3]][champion_li
    numberLosesCombination[champion_list[5]][champion_list[6]] += 1
```

<numberWinsChampion>

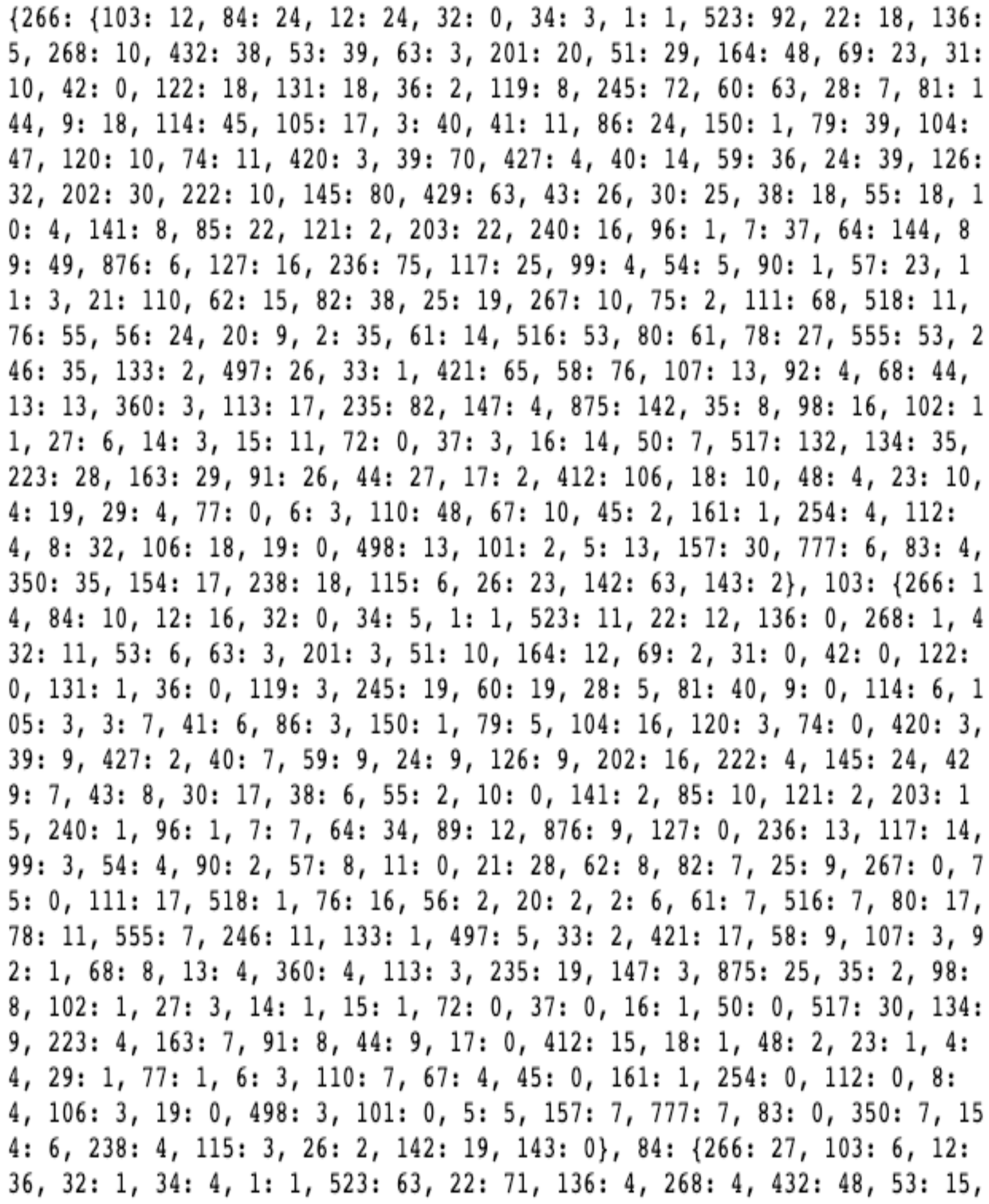
챔피언 vs 상대팀 챔피언(5명 각각) 승리횟수 counting

<numberWinsCombination>

챔피언 + 아군챔피언1, 챔피언 + 아군챔피언2 ... 챔피언 + 아군챔피언4 승리횟수 counting

<numberLosesCombination>

챔피언 + 아군챔피언1, 챔피언 + 아군챔피언2 ... 챔피언 + 아군챔피언4 패배횟수 counting



```
{266: {103: 12, 84: 24, 12: 24, 32: 0, 34: 3, 1: 1, 523: 92, 22: 18, 136:
5, 268: 10, 432: 38, 53: 39, 63: 3, 201: 20, 51: 29, 164: 48, 69: 23, 31:
10, 42: 0, 122: 18, 131: 18, 36: 2, 119: 8, 245: 72, 60: 63, 28: 7, 81: 1
44, 9: 18, 114: 45, 105: 17, 3: 40, 41: 11, 86: 24, 150: 1, 79: 39, 104:
47, 120: 10, 74: 11, 420: 3, 39: 70, 427: 4, 40: 14, 59: 36, 24: 39, 126:
32, 202: 30, 222: 10, 145: 80, 429: 63, 43: 26, 30: 25, 38: 18, 55: 18, 1
0: 4, 141: 8, 85: 22, 121: 2, 203: 22, 240: 16, 96: 1, 7: 37, 64: 144, 8
9: 49, 876: 6, 127: 16, 236: 75, 117: 25, 99: 4, 54: 5, 90: 1, 57: 23, 1
1: 3, 21: 110, 62: 15, 82: 38, 25: 19, 267: 10, 75: 2, 111: 68, 518: 11,
76: 55, 56: 24, 20: 9, 2: 35, 61: 14, 516: 53, 80: 61, 78: 27, 555: 53, 2
46: 35, 133: 2, 497: 26, 33: 1, 421: 65, 58: 76, 107: 13, 92: 4, 68: 44,
13: 13, 360: 3, 113: 17, 235: 82, 147: 4, 875: 142, 35: 8, 98: 16, 102: 1
1, 27: 6, 14: 3, 15: 11, 72: 0, 37: 3, 16: 14, 50: 7, 517: 132, 134: 35,
223: 28, 163: 29, 91: 26, 44: 27, 17: 2, 412: 106, 18: 10, 48: 4, 23: 10,
4: 19, 29: 4, 77: 0, 6: 3, 110: 48, 67: 10, 45: 2, 161: 1, 254: 4, 112:
4, 8: 32, 106: 18, 19: 0, 498: 13, 101: 2, 5: 13, 157: 30, 777: 6, 83: 4,
350: 35, 154: 17, 238: 18, 115: 6, 26: 23, 142: 63, 143: 2}, 103: {266: 1
4, 84: 10, 12: 16, 32: 0, 34: 5, 1: 1, 523: 11, 22: 12, 136: 0, 268: 1, 4
32: 11, 53: 6, 63: 3, 201: 3, 51: 10, 164: 12, 69: 2, 31: 0, 42: 0, 122:
0, 131: 1, 36: 0, 119: 3, 245: 19, 60: 19, 28: 5, 81: 40, 9: 0, 114: 6, 1
05: 3, 3: 7, 41: 6, 86: 3, 150: 1, 79: 5, 104: 16, 120: 3, 74: 0, 420: 3,
39: 9, 427: 2, 40: 7, 59: 9, 24: 9, 126: 9, 202: 16, 222: 4, 145: 24, 42
9: 7, 43: 8, 30: 17, 38: 6, 55: 2, 10: 0, 141: 2, 85: 10, 121: 2, 203: 1
5, 240: 1, 96: 1, 7: 7, 64: 34, 89: 12, 876: 9, 127: 0, 236: 13, 117: 14,
99: 3, 54: 4, 90: 2, 57: 8, 11: 0, 21: 28, 62: 8, 82: 7, 25: 9, 267: 0, 7
5: 0, 111: 17, 518: 1, 76: 16, 56: 2, 20: 2, 2: 6, 61: 7, 516: 7, 80: 17,
78: 11, 555: 7, 246: 11, 133: 1, 497: 5, 33: 2, 421: 17, 58: 9, 107: 3, 9
2: 1, 68: 8, 13: 4, 360: 4, 113: 3, 235: 19, 147: 3, 875: 25, 35: 2, 98:
8, 102: 1, 27: 3, 14: 1, 15: 1, 72: 0, 37: 0, 16: 1, 50: 0, 517: 30, 134:
9, 223: 4, 163: 7, 91: 8, 44: 9, 17: 0, 412: 15, 18: 1, 48: 2, 23: 1, 4:
4, 29: 1, 77: 1, 6: 3, 110: 7, 67: 4, 45: 0, 161: 1, 254: 0, 112: 0, 8:
4, 106: 3, 19: 0, 498: 3, 101: 0, 5: 5, 157: 7, 777: 7, 83: 0, 350: 7, 15
4: 6, 238: 4, 115: 3, 26: 2, 142: 19, 143: 0}, 84: {266: 27, 103: 6, 12:
36, 32: 1, 34: 4, 1: 1, 523: 63, 22: 71, 136: 4, 268: 4, 432: 48, 53: 15,
```

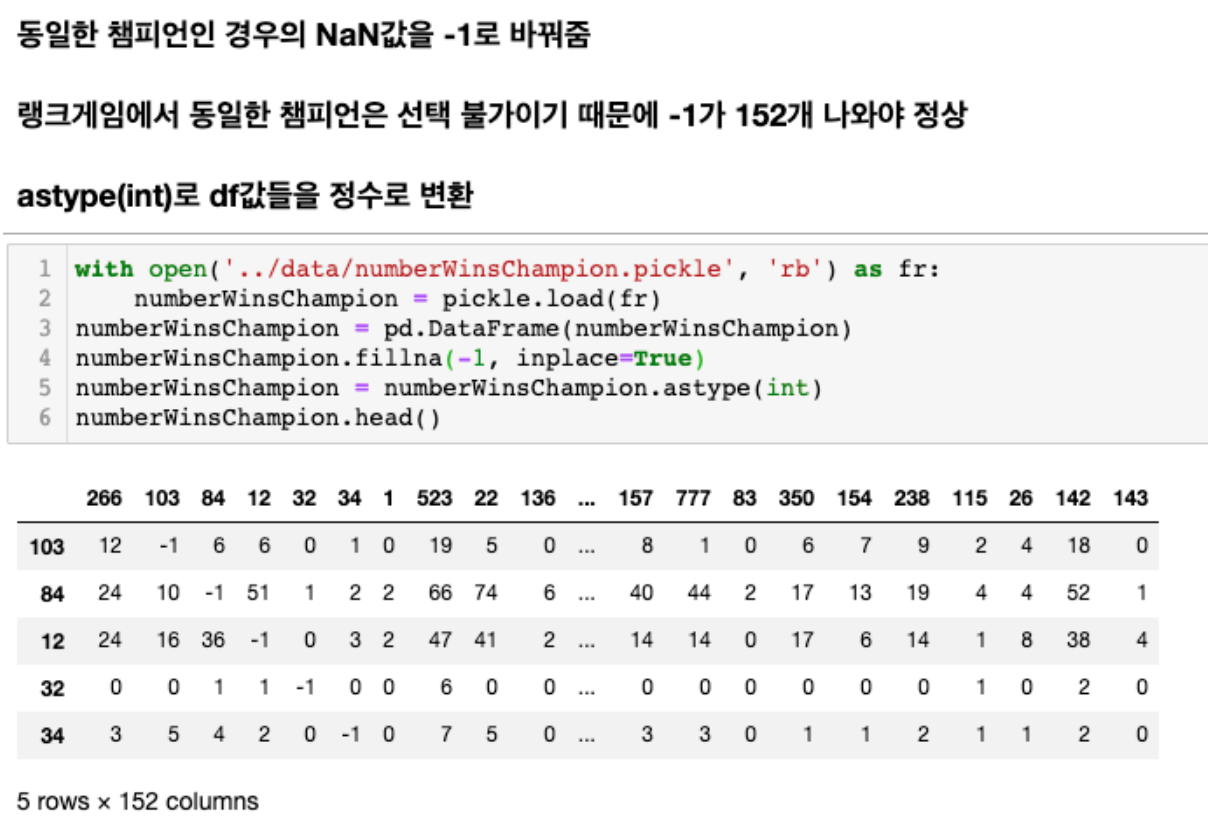
numberWinsChampion 딕셔너리의 예시

ex)

Key = 해당챔피언id

value의 type : 딕셔너리

{ 1번째 챔피언 id : 승리횟수 , … , 151번째 챔피언 id : 승리횟수}



동일한 챔피언인 경우의 NaN값을 -1로 바꿔줌

랭크게임에서 동일한 챔피언은 선택 불가이기 때문에 -1가 152개 나와야 정상

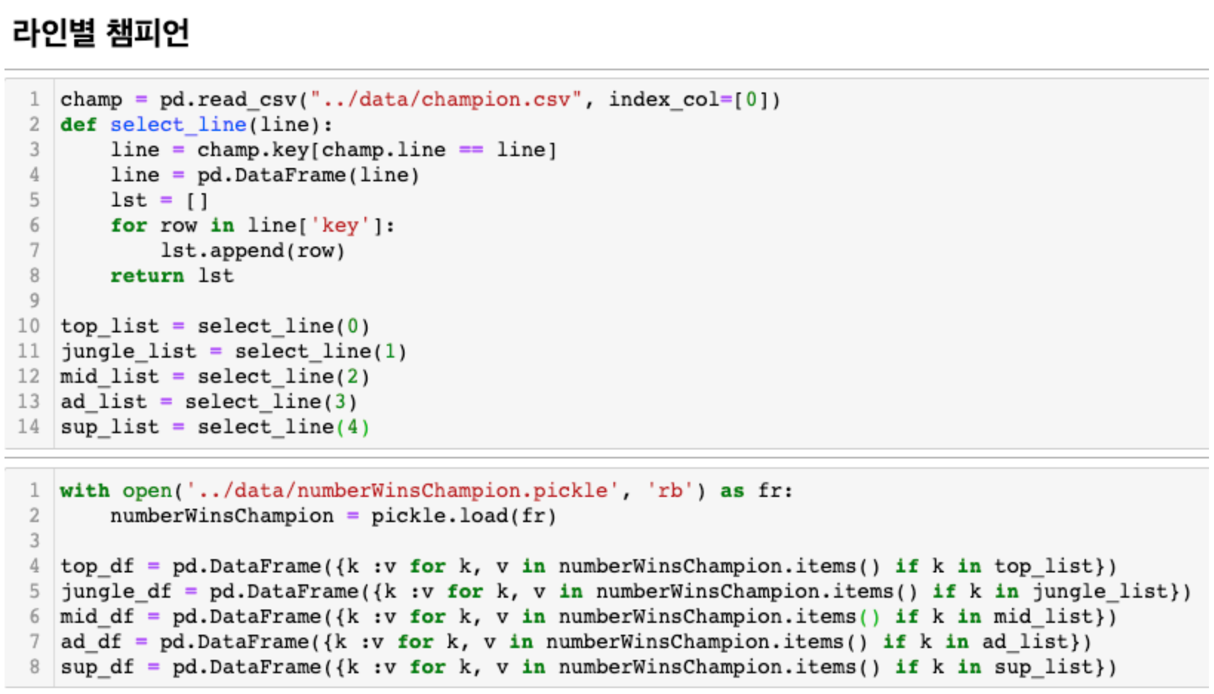
astype(int)로 df값들을 정수로 변환

```python
with open('../data/numberWinsChampion.pickle', 'rb') as fr:
    numberWinsChampion = pickle.load(fr)
numberWinsChampion = pd.DataFrame(numberWinsChampion)
numberWinsChampion.fillna(-1, inplace=True)
numberWinsChampion = numberWinsChampion.astype(int)
numberWinsChampion.head()
```

	266	103	84	12	32	34	1	523	22	136	...	157	777	83	350	154	238	115	26	142	143
103	12	-1	6	6	0	1	0	19	5	0	...	8	1	0	6	7	9	2	4	18	0
84	24	10	-1	51	1	2	2	66	74	6	...	40	44	2	17	13	19	4	4	52	1
12	24	16	36	-1	0	3	2	47	41	2	...	14	14	0	17	6	14	1	8	38	4
32	0	0	1	1	-1	0	0	6	0	0	...	0	0	0	0	0	0	1	0	2	0
34	3	5	4	2	0	-1	0	7	5	0	...	3	3	0	1	1	2	1	1	2	0

5 rows × 152 columns

<위의 딕셔너리를 DataFrame 형태로(numberWinsCombination도 같은 형태)>



라인별 챔피언

```python
champ = pd.read_csv("../data/champion.csv", index_col=[0])
def select_line(line):
    line = champ.key[champ.line == line]
    line = pd.DataFrame(line)
    lst = []
    for row in line['key']:
        lst.append(row)
    return lst

top_list = select_line(0)
jungle_list = select_line(1)
mid_list = select_line(2)
ad_list = select_line(3)
sup_list = select_line(4)
```

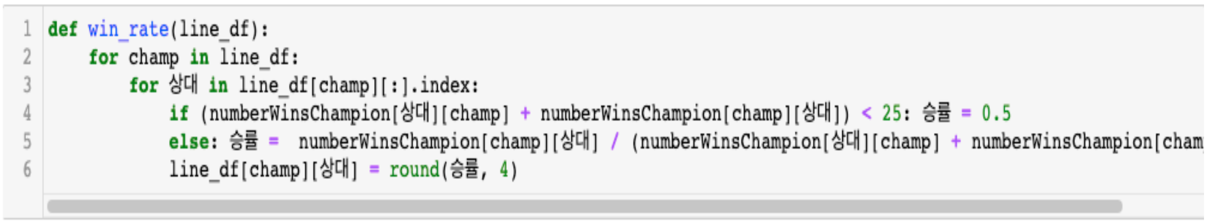
```python
with open('../data/numberWinsChampion.pickle', 'rb') as fr:
    numberWinsChampion = pickle.load(fr)

top_df = pd.DataFrame({k :v for k, v in numberWinsChampion.items() if k in top_list})
jungle_df = pd.DataFrame({k :v for k, v in numberWinsChampion.items() if k in jungle_list})
mid_df = pd.DataFrame({k :v for k, v in numberWinsChampion.items() if k in mid_list})
ad_df = pd.DataFrame({k :v for k, v in numberWinsChampion.items() if k in ad_list})
sup_df = pd.DataFrame({k :v for k, v in numberWinsChampion.items() if k in sup_list})
```

<위의 데이터 프레임을 5개의 포지션 별로 쪼갬; 5개의 데이터 프레임 생성>

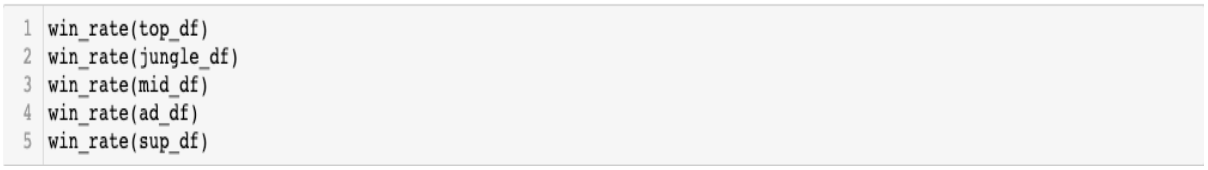
라인별 DataFrame

columns -> 해당 라인 챔피언들

rows -> 전체 챔피언들



```python
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
            else: 승률 =  numberWinsChampion[champ][상대] / (numberWinsChampion[상대][champ] + numberWinsChampion[cham
            line_df[champ][상대] = round(승률, 4)
```



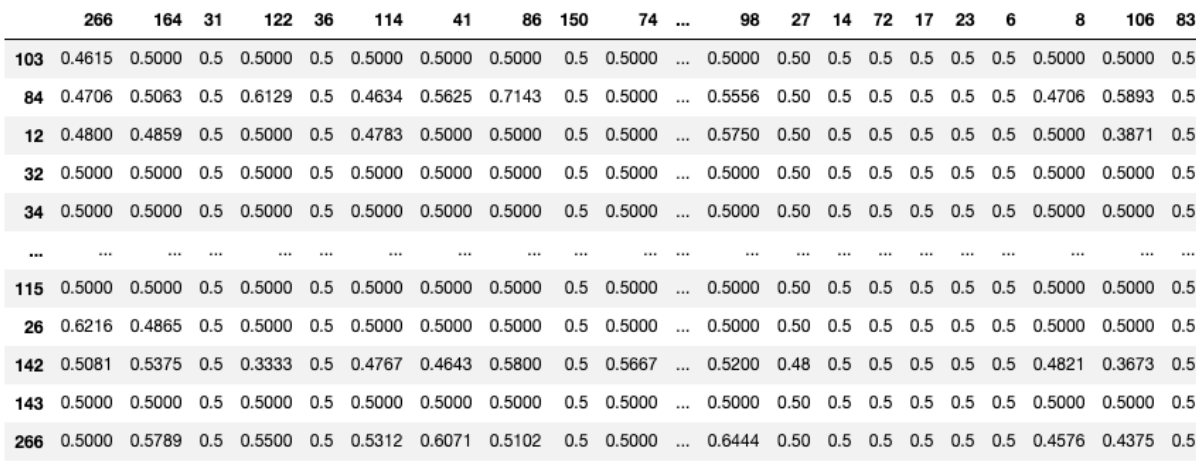
```python
win_rate(top_df)
win_rate(jungle_df)
win_rate(mid_df)
win_rate(ad_df)
win_rate(sup_df)
```

A챔피언 vs B챔피언의 경기 횟수가 25경기 보다 작으면, 승률 = 50%

A승률 : A승리 / A승리 + B승리

B승률 : B승리 / A승리 + B승리

위의 함수를 각 라인별DataFrame에 적용



	266	164	31	122	36	114	41	86	150	74	...	98	27	14	72	17	23	6	8	106	83
103	0.4615	0.5000	0.5	0.5000	0.5	0.5000	0.5000	0.5000	0.5	0.5000	...	0.5000	0.50	0.5	0.5	0.5	0.5	0.5	0.5000	0.5000	0.5
84	0.4706	0.5063	0.5	0.6129	0.5	0.4634	0.5625	0.7143	0.5	0.5000	...	0.5556	0.50	0.5	0.5	0.5	0.5	0.5	0.4706	0.5893	0.5
12	0.4800	0.4859	0.5	0.5000	0.5	0.4783	0.5000	0.5000	0.5	0.5000	...	0.5750	0.50	0.5	0.5	0.5	0.5	0.5	0.5000	0.3871	0.5
32	0.5000	0.5000	0.5	0.5000	0.5	0.5000	0.5000	0.5000	0.5	0.5000	...	0.5000	0.50	0.5	0.5	0.5	0.5	0.5	0.5000	0.5000	0.5
34	0.5000	0.5000	0.5	0.5000	0.5	0.5000	0.5000	0.5000	0.5	0.5000	...	0.5000	0.50	0.5	0.5	0.5	0.5	0.5	0.5000	0.5000	0.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
115	0.5000	0.5000	0.5	0.5000	0.5	0.5000	0.5000	0.5000	0.5	0.5000	...	0.5000	0.50	0.5	0.5	0.5	0.5	0.5	0.5000	0.5000	0.5
26	0.6216	0.4865	0.5	0.5000	0.5	0.5000	0.5000	0.5000	0.5	0.5000	...	0.5000	0.50	0.5	0.5	0.5	0.5	0.5	0.5000	0.5000	0.5
142	0.5081	0.5375	0.5	0.3333	0.5	0.4767	0.4643	0.5800	0.5	0.5667	...	0.5200	0.48	0.5	0.5	0.5	0.5	0.5	0.4821	0.3673	0.5
143	0.5000	0.5000	0.5	0.5000	0.5	0.5000	0.5000	0.5000	0.5	0.5000	...	0.5000	0.50	0.5	0.5	0.5	0.5	0.5	0.5000	0.5000	0.5
266	0.5000	0.5789	0.5	0.5500	0.5	0.5312	0.6071	0.5102	0.5	0.5000	...	0.6444	0.50	0.5	0.5	0.5	0.5	0.5	0.4576	0.4375	0.5

top_DataFrame (탑 라인 승률df 의 예시)

문제점]

- 한 챔피언만 숙련도가 높은 유저의 데이터를 제거를 할 생각이었는데
 90,000게임 x 10명의 유저 각각의 데이터를 조사 하는 것이   900,000개의 데이터를
Riot API에 요청을 해야하는데 2분에 100개를 제공받는 상황을 감안하면 현실적으로
데이터를 다 받는 것이 불가능 하다고 판단되어서 모두 포함시키기로 했다.

3. 진행 예정사항

1) 챔피언 승률, 아군 챔피언 조합 승률을 모델에 맞게 Scaling
2) 적합한 모델 찾기
3) 자신의 Most Play 챔피언 데이터 받아오는 과정 구현
4) tkiniter를 활용한 GUI 개발 시작