RIOT API를 활용한 라인별 챔피언 승률 예측

1. 개요

- 파이썬 수업 중 리그오브레전드 게임사 RIOT에서 제공해주는 데이터가 있다는 것을 알게 되었고, RIOT API를 통해 분석 프로젝트를 수행하고자 해당 분석을 시작하였다.

2. 분석의 방향성

- 캐릭터별 상성은 필히 존재하며 이런 캐릭터별 상성으로 인해 라인전이 달라진다. 따라서 A가 같은 캐릭터를 해도 상대가 무슨 캐릭터를 하냐에 따라 초반 플레이가 달라지고 결과에 반영된다 이처럼 캐릭터별 라인전 데이터가 게임에 영향을 끼치는 정도와 이를 통한 승률 예측을 해보는 것이다.

3. 기준

1) 캐릭터 : 롤에는 160여개의 캐릭터가 존재하며 모든 챔프가 모든라인에서 등장할 수 있다.
허나Values_Count() 를 활용하여 15개 이하는 "비주류 캐릭터"로 분류하여 데이터에서 제외한다.

2) 포지션 : 1,6탑/2,7정글/3,8미드/4,9원딜/5,10서폿으로 배정되오나 게임 내에서 스위칭을 한 변수는 배제한다

3) 라인전 : RIOT API에서 제공하는 TimeLine Match Data를 통해 15분 이내의 데이터를 라인전 데이터로 규정 한다.

4) 분석기법 : 승 또는 패를 구분하기에 LogisticRegression을 선택한다.

4. 분석 순서

1) API 호출
2) DB 적재
3) 데이터 전처리
4) 데이터 분석
5) 결과 도출

5. 분석환경

1) DB : MYSQL 8.0
2) 사용언어 : PYTHON / SQL
3) 활용 도구 : Jupyter
4) 사용 라이브러리 : pandas, Sklearn 등

6. 분석목표

- 라인 각각 정확도 70%

API 호출 (Jupyter 활용)

1. User의 게임 닉네임 추출 후 DB 적재

- MySQL Connect 객체 생성
- 라이엇에서 제공해주는 url 및 header 작성
- Json으로 반환된 결과 중 필요한 데이터만 선택
- 선택한 데이터를 summerInfo 테이블에 insert
- DB 연결 해제

```
# 서머너 네임 db적재
con = pymysql.connect(host=[redacted], user='root', password='root1234',db='riot', charset='utf8')
cur = con.cursor()
num = 1

for save in tqdm(range(100)):
    api_key = "RGAPI-6f4eed01-264c-4333-8a96-fcdafdf28727"
    url = "https://kr.api.riotgames.com/lol/league/v4/entries/RANKED_SOLO_5x5/PLATINUM/II?page="+str(num)
    header = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36",
    "Accept-Language": "ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7",
    "Accept-Charset": "application/x-www-form-urlencoded; charset=UTF-8",
    "Origin": "https://developer.riotgames.com",
    "X-Riot-Token": "RGAPI-6f4eed01-264c-4333-8a96-fcdafdf28727"
    }
    res = req.get(url, headers=header)
    summerner = res.json()
    num += 3
    time.sleep(2)
    for i in summerner:
        summonerName = i['summonerName']
#         print(summonerName)
        sql = "INSERT INTO summerinfo (summernerName) VALUES (%s)"
        val = (summonerName)
        cur.execute(sql,val)
    con.commit()
cur.close()
con.close()
```

그림 01 : 서머너 네임 DB에 적재

sno	summernerName	puuids
1	야 Theshy	wLN3njg8mGAgRPQxx0dHZF25z-DGEZ0WOBVD...
2	응애소나	m1IUB-D5k35egcyjYc2VkiYQzLtCQK1q7Q1uNAf...
4	쿵 재	3DCZ3VSTAWyD9e17ZqpuRRpz5jMQsbDJ6Gwu...
5	기분좋게학내기	AddBp1biqAeqSlNcyL80p7RU4bcIqdoWfK-gCRn...
7	Serpiente	JvNJu9peZi5gPBBSSRibH5-azIpC_RhIOOLkx-G6...
8	소금문	S9N5k7FohHXINfTMLuBHb2JYc9d2Wa-5Mfj-Lfv...
9	부먹 민초 한입충	jqY3CWg8ytJcVa0naVyhcg8xj-FFMgCWcSupm5...
10	오면 더블킬	jjul67DwSZ97tZR3EglHWSrOkQuTdgIvrlLdd8QS...
11	애기헤스	jry3-frfIpBwJ3Z0aAP-ANArsw7AUwtSAFClZ6fs...
12	karmx	qkhpdx5llLAE-b7SZDUW38wkQbv1kDnIykcvAC-...
13	정글몹에죽는사람	2qzSCaP7XGL52iQHXMNG6IU82dUuyGpersqa6-...
14	아뜨앙	qFJng53IFp35fyx0BdiEshMo46ePmuzKmNwN4S...
15	아니 나 킬 좀 줘	1Cl68CSajlO8SAwCYQqJG2N1oIiFYMR9tZe756...
16	도 끼 맞 아 라	dx799-W6X03ColuK2cTjCedUj-sclXdfTtv75dRQs...
17	복현동여부초밥	iuvZuEIa9751528ZsnC8EsLgbKaqgTuqDMwFI4R...
19	Ciara bravo	xA4Lb5Y5qy1N1hs4lZEyMxVllp54rjdZ5CaNJ2fn...
20	에코 시간 역행	ARSOdtHm8fsr9qzJKrcGF8--1-Kopxh82j6kuMXR...
21	탱크놓치면근손실	_4_xjlP07EvM4rZdwG_gNt2f1cZqndD3MbNWny...
22	Lam thu di	7wUW52Dtr7kI1R51nB3YzcM6KMD9xkdn7S3F2...

그림 02 : summerInfo 테이블 데이터

API 호출

2. 추출한 User의 게임 닉네임을 통해 고유 puuid 추출 후 DB 적재

- RIOT API를 이용하여 나온 Json 객체를 리턴해주는 함수 정의
- MySQL Connect 객체 생성
- summerInfo 테이블에의 닉네임 데이터로 for문 작성
- searchId 함수를 통과해서 나온 json객체 내 필요한 데이터를 summerInfo에 업데이트

```python
# 소환사이름으로 puuid 추출함수
def searchId(x) :
    name = x
    enName = parse.quote(name)
#     print(enName)
    api_key = "RGAPI-1ecac1ec-d6b6-40ca-ac71-df39b3114067"
    url = "https://kr.api.riotgames.com/lol/summoner/v4/summoners/by-name/"+enName
    header = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36",
        "Accept-Language": "ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7",
        "Accept-Charset": "application/x-www-form-urlencoded; charset=UTF-8",
        "Origin": "https://developer.riotgames.com",
        "X-Riot-Token": api_key
    }
    res = req.get(url, headers=header)
    search = res.json()
    return search
```

그림 03 : puuid를 추출하기 위한 api호출 양식을 함수로 정의

```python
# puuid 추출후 db에 적재
con = pymysql.connect(host=[redacted], user='root', password='root1234',db='riot', charset='utf8')
cur = con.cursor()
sel_sql = "SELECT summernerName FROM summerinfo"
cur.execute(sel_sql)
summerner = cur.fetchall()
for l in tqdm(summerner):
    name = l[0]
    try:
        res = searchId(name)
        puuid = res.get('puuid')
        sql = "UPDATE summerinfo SET puuids= %s where summernerName = %s"
        val = (puuid,name)
        cur.execute(sql,val)
        time.sleep(2)
    except Exception as e:
        print("닉네임"+l+"에서 에외발생")
        print(e)
        time.sleep(120)
    con.commit()
con.close()
```

그림 04 : 그림 03의 함수를 이용하여 puuid를 summerInfo 테이블에 UPDATE

sno	summernerName	puuids
1	야 Theshy	wLN3njg8mGAgRPQxx0dHZF25z-DGEZ0WOBVD...
2	응애소나	m1IUB-D5k35egcyjYc2VkiYQzLtCQK1q7Q1uNAf...
4	쿵 재	3DCZ3VSTAWyD9e17ZqpuRRpz5jMQsbDJ6Gwu...
5	기분좋게화내기	AddBp1biqAeqSlNcyL80p7RU4bcIqdoWfK-gCRn...
7	Serpiente	JvNJu9peZi5gPBBSSRibH5-azIpC_RhIOOLkx-G6...
8	소금몬	S9N5k7FohHXINfTMLuBHb2JYc9d2Wa-5Mfj-Lfv...
9	부먹 민초 한입충	jqY3CWg8ytJcVa0naVyhcg8xj-FFMgCWcSupm5...
10	오면 더블킬	jjul67DwSZ97tZR3EglHWSrOkQuTdgIvrlLdd8QS...
11	애기헤스	jry3-frfIpBwJ3Z0aAP-ANArsw7AUwtSAFClZ6fs...
12	karmx	qkhpdx5llLAE-b7SZDUW38wkQbv1kDnIykcvAC-...
13	정글몹에죽는사람	2qzSCaP7XGL52iQHXMNG6IU82dUuyGpersqa6-...
14	아뜨앙	qFJng53IFp35fyx0BdiEshMo46ePmuzKmNwN4S...
15	아니 나 킬 좀 줘	1Cl68CSajlO8SAwCYQqJG2N1oIiFYMR9tZe756...
16	도 끼 맞 아 라	dx799-W6X03ColuK2cTjCedUj-sdXdfTtv75dRQs...
17	복현동여부초밥	iuvZuEIa9751528ZsnC8EsLgbKaqgTuqDMwFI4R...
19	Ciara bravo	xA4Lb5Y5qy1N1hs4lZEyMxVllp54rjdZ5CaNJ2fn...
20	에코 시간 역행	ARSOdtHm8fsr9qzJKrcGF8--1-Kopxh82j6kuMXR...
21	탱크놀지면근손실	_4_xjlP07EvM4rZdwG_gNt2f1cZqndD3MbNWny...
22	Lam thu di	7wUW52Dtr7kI1R51nB3YzcM6KMD9xkdn7S3F2...

그림 05 : summerInfo 테이블 데이터

API 호출

3. puuid로 matchId 추출 후 DB에 적재

- MySQL Connect 객체 생성
- summerInfo 테이블의 puuids 데이터를 조회
- 양식에 맞게 api 호출에 필요한 파라미터 작성
- matchId 테이블에 응답된 데이터 적재

```
# puuid로 22시즌 랭크매치데이터 가져오기
con = pymysql.connect(host=[redacted], user='root', password='root1234',db='riot', charset='utf8')
cur = con.cursor()
sql = "SELECT puuids FROM summerinfo"
cur.execute(sql)
match = cur.fetchall()
api_key = "RGAPI-1ecac1ec-d6b6-40ca-ac71-df39b3114067"
start= str(int(datetime(2022, 1, 8).replace(tzinfo=timezone.utc).timestamp()))
end = str(int(datetime(2022, 11, 9).replace(tzinfo=timezone.utc).timestamp()))
for m in tqdm(match):
#     print(m[0])
    startCount = 0
    totalCount = 100
    url = 'https://asia.api.riotgames.com/lol/match/v5/matches/by-puuid/'+str(m[0])+'/ids?startTime='+start+'&endTime='+end+'&type=ranked&start='+str(startCount)+'&count='+str(totalCount)
    header = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36",
        "Accept-Language": "ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7",
        "Accept-Charset": "application/x-www-form-urlencoded; charset=UTF-8",
        "Origin": "https://developer.riotgames.com",
        "X-Riot-Token": api_key
    }
    try :
        res = req.get(url, headers=header)
        matchId = res.json()
        time.sleep(2)
        for index, i in enumerate(matchId) :
            insert = "INSERT INTO matchId (matchId) VALUES ( %s )"
            val = i
            cur.execute(insert, val)
    except Exception as e:
        print("puuid"+i+"의"+index+" 행에서 에외발생")
        print(e)
        time.sleep(120)
    con.commit()
cur.close()
con.close()
```

그림 06 : summerInfo 테이블의 puuids 데이터를 통해 matchId를 구해오는 코드

	mno	matchId
▶	23000	KR_5874572778
	23001	KR_5874539126
	23002	KR_5874525274
	23003	KR_5874443665
	23004	KR_5874379021
	23005	KR_5874374082
	23006	KR_5874266555
	23007	KR_5874270306
	23008	KR_5873895861
	23009	KR_5873828676
	23010	KR_5873835463
	23011	KR_5866644634
	23012	KR_5866632310
	23013	KR_5866650061
	23014	KR_5861954249
	23015	KR_5861877873
	23016	KR_5861892261
	23017	KR_5861824608
	23018	KR_5861716436
	23019	KR_5861641884

그림 07 : matchId 테이블 데이터

API 호출

4. matchId로 게임데이터 추출 후 적재

4.1 사전준비

- MySQL Connect 객체 생성
- 데이터를 추가할 테이블을 DataFrame 형태로 변환
- 코드에 활용될 변수 정의

```
# DB테이블 DataFrame으로 변경
con = pymysql.connect(host='127.0.0.1', user='root', password='root1234',db='riot', charset='utf8')
# cur = con.cursor()
#cur.execute("select * from addata")
import pandas as pd
topdata = pd.read_sql("describe topdata", con, index_col='Field')
jgdata = pd.read_sql("describe jgdata", con, index_col='Field')
middata = pd.read_sql("describe middata", con, index_col='Field')
addata = pd.read_sql("describe addata", con, index_col='Field')
supdata = pd.read_sql("describe supdata", con, index_col='Field')
```

그림 08 : 각 포지션 테이블을 DataFrame으로 변환

```
#15분 이전 블루팀 스펙(insert)챔프이름,블루킬,블루골드,블루딜량,블루탱량,블루렙,블루미니언,승리
insert = "INSERT INTO %s (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s) VALUES (%s,'%s',%s,%s,%s,%s,%s,%s,%s)"
#15분 이전 레드팀 스펙(update) 테이블명,챔프이름,레드킬,레드골드,레드딜량,레드탱량,레드렙,레드미니언,no
update = "UPDATE %s SET %s='%s',%s=%d,%s=%d,%s=%s,%s=%s,%s=%d,%s=%s WHERE %s=%s"

#DB테이블의 no를 유동적으로 변경하기 위해 리스트로 선언
noList = ['tno','jno','mno','ano','sno']

#DB테이블 명을 유동적으로 변경하기 위해 리스트로 선언
position = ['topdata','jgdata','middata','addata','supdata']

#챔피언명 리스트로반환
champList = []

#포지션별 no 인덱스
tindex=jindex=mindex=aindex=sindex = 29550

#킬count를 담을 변수 선언
bt=bj=bm=ba=bs=rt=rj=rm=ra=rs = 0

#승리팀(True,False) 구분
winList = []
winSep = 3
```

그림 09 : 하기 코드에 사용될 변수 정의

API 호출

4.2 코드 구현

- 오리지널 게임데이터 호출 코드 작성(orgData)
- 타임라인 게임데이터 호출 코드 작성(timeData)
- orgData 내 필요한 부분을 sampledata로 정의
- timeData의 15분 캐릭터 스펙을 sampledata2로 정의
- sampledata에서 승패여부와 사용된 캐릭터 추출

```
for matchId in tqdm(getMatchId):
#     print(type(matchId[0]))
#     일반매치데이터 호출
    try:
        orgUrl = 'https://asia.api.riotgames.com/lol/match/v5/matches/'+matchId[0]+'?api_key='+api_key
        orgMatch = req.get(orgUrl, headers=header)
        orgData = orgMatch.json()

        #타임라인매치데이터 호출
        timeUrl = 'https://asia.api.riotgames.com/lol/match/v5/matches/'+matchId[0]+'/timeline?api_key='+api_key
        timeMatch = req.get(timeUrl,headers=header)
        timeData = timeMatch.json()
    except Exception as e:
        print(matchId[0]+" 호출중 예외발생")
        print(e)
        time.sleep(5)
    try:
        sampledata = orgData['info']['participants']
        if len(timeData['info']['frames']) > 15:
            sampledata2 = timeData['info']['frames'][15]['participantFrames']
            #데이터전처리 후 db적재
            for winResult in sampledata:
                winList.append(winResult['win'])
                if winList[0] == True:
                    winSep = 0
                else :
                    winSep = 1
            #     print(winSep)
            for champName in sampledata:
            #     print(champName['championName'])
                champList.append(champName['championName'])
```

그림 10 : API호출,승패여부,사용캐릭터 추출 코드

- 15분동안 각 포지션별 Kill score를 계산

```
for timeline in range(1,16):
    for count in timeData['info']['frames'][timeline]['events']:
        if count.get('type') == 'CHAMPION_KILL':
#             print(count['killerId'])
            if count['killerId'] == 1:
                bt+=1
            if count['killerId'] == 2:
                bj+=1
            if count['killerId'] == 3:
                bm+=1
            if count['killerId'] == 4:
                ba+=1
            if count['killerId'] == 5:
                bs+=1
            if count['killerId'] == 6:
                rt+=1
            if count['killerId'] == 7:
                rj+=1
            if count['killerId'] == 8:
                rm+=1
            if count['killerId'] == 9:
                ra+=1
            if count['killerId'] == 10:
                rs+=1
```

그림 11 : 각 포지션별 kill score 계산

API 호출

4.2 코드 구현

- 1~5번 소환사는 insert문을 통해 데이터 적재(블루팀)
- 6~10번 소환사는 update문을 통해 데이터 적재(레드팀)

```
for i in range(1,11):
    if i == 1:
#           print('블루탑')#no,챔프이름,블루킬,블루골드,블루딜량,블루탱량,블루랩,블루미니언,승리
        bt_level = sampledata2[str(i)]['level']
        bt_minionsKilled = sampledata2[str(i)]['minionsKilled']
        bt_totalGold = sampledata2[str(i)]['totalGold']
        bt_totalDamageDoneToChampions = sampledata2[str(i)]['damageStats']['totalDamageDoneToChampions']
        bt_totalDamageTaken = sampledata2[str(i)]['damageStats']['totalDamageTaken']
        var_bt = (position[0],noList[0],topdata.index[1],topdata.index[2],topdata.index[3],topdata.index[4],topdata.index[5],topdata.index[6],topdata.index[7],topdata.index[15],
            tindex,champList[0],bt,bt_totalGold,bt_totalDamageDoneToChampions,bt_totalDamageTaken,bt_level,bt_minionsKilled,winSep)
        print(var_bt)
        cur.execute(insert % var_bt)
    if i == 2:
#           print('블루정글')
        bj_level = sampledata2[str(i)]['level']
        bj_minionsKilled = sampledata2[str(i)]['minionsKilled']
        bj_totalGold = sampledata2[str(i)]['totalGold']
        bj_totalDamageDoneToChampions = sampledata2[str(i)]['damageStats']['totalDamageDoneToChampions']
        bj_totalDamageTaken = sampledata2[str(i)]['damageStats']['totalDamageTaken']
        var_bj = (position[1],noList[1],jgdata.index[1],jgdata.index[2],jgdata.index[3],jgdata.index[4],jgdata.index[5],jgdata.index[6],jgdata.index[7],jgdata.last_valid_index(),
            jindex,champList[1],bj,bj_totalGold,bj_totalDamageDoneToChampions,bj_totalDamageTaken,bj_level,bj_minionsKilled,winSep)
        print(var_bj)
        cur.execute(insert % var_bj)
```

그림 12 : 블루팀 데이터 적재 코드

```
    if i == 6: #'UPDATE %s SET %s=%s,%s=%s,%s=%s,%s=%s,%s=%s,%s=%s,%s=%s,%s=%s WHERE %s=%s' 테이블명,챔프이름,레드킬,레드골드,레드딜량,레드탱량,레드랩,레드미니언,no
#           print('레드탑')
        rt_level = sampledata2[str(i)]['level']
        rt_minionsKilled = sampledata2[str(i)]['minionsKilled']
        rt_totalGold = sampledata2[str(i)]['totalGold']
        rt_totalDamageDoneToChampions = sampledata2[str(i)]['damageStats']['totalDamageDoneToChampions']
        rt_totalDamageTaken = sampledata2[str(i)]['damageStats']['totalDamageTaken']
        var_rt = (position[0],topdata.index[8],champList[5],topdata.index[9],rt,topdata.index[10],rt_totalGold,topdata.index[11],rt_totalDamageDoneToChampions,
            topdata.index[12],rt_totalDamageTaken,topdata.index[13],rt_level,topdata.index[14],rt_minionsKilled,noList[0],tindex)
        print(var_rt)
        cur.execute(update % var_rt)
    if i == 7:
#           print('레드정글')
        rj_level = sampledata2[str(i)]['level']
        rj_minionsKilled = sampledata2[str(i)]['minionsKilled']
        rj_totalGold = sampledata2[str(i)]['totalGold']
        rj_totalDamageDoneToChampions = sampledata2[str(i)]['damageStats']['totalDamageDoneToChampions']
        rj_totalDamageTaken = sampledata2[str(i)]['damageStats']['totalDamageTaken']
        var_rj = (position[1],jgdata.index[8],champList[6],jgdata.index[9],rj,jgdata.index[10],rj_totalGold,jgdata.index[11],rt_totalDamageDoneToChampions,
            jgdata.index[12],rj_totalDamageTaken,jgdata.index[13],rj_level,jgdata.index[14],rj_minionsKilled,noList[1],jindex)
        print(var_rj)
        cur.execute(update % var_rj)
```

그림 13 : 레드팀 데이터 적재 코드

tno	blue_top_champ	bt_kill	bt_gold	bt_damageDone	bt_damageTaken	bt_level	bt_minion	red_top_champ	rt_kill	rt_gold	rt_damageDone	rt_damageTaken	rt_level	rt_minion	result
1	Gangplank	1	5256	11067	12651	10	93	Fiora	3	4944	11610	15583	11	90	1
2	Zac	3	5987	9122	12814	11	114	Renekton	0	5342	8326	13051	10	125	0
3	Akali	1	5486	4786	5463	11	129	Fiora	0	4259	3132	6348	10	112	0
4	Aatrox	3	5633	6102	9976	10	91	Jax	1	4087	5130	11299	9	50	1
5	Sejuani	4	6880	7338	9899	11	119	Nocturne	1	5083	5966	8925	10	84	1
6	Tryndamere	4	6296	11986	17954	12	103	Camille	3	5323	12294	12931	11	91	1
7	Fiora	2	4831	5916	10571	11	98	Ryze	2	5025	8323	7069	10	97	1
8	Ryze	1	4958	8135	6567	11	106	Olaf	0	4215	5199	11657	10	94	0
9	Khazix	2	5470	8889	8065	11	97	Trundle	1	5030	2687	9807	9	7	1
10	Ornn	0	4107	5872	11046	10	102	Gnar	5	7044	11872	8808	12	130	1
11	Quinn	3	5639	5147	5724	10	75	Shyvana	2	6772	3849	7220	11	123	1
12	Nasus	1	4916	2991	7003	11	113	Kayle	0	5230	1958	3560	10	128	1
13	Aatrox	4	7221	9003	10125	11	109	Fiora	0	3881	3452	7528	9	82	0
14	Akshan	1	4806	7506	8149	10	85	Lillia	1	5182	4447	7111	11	99	0
15	Shen	0	3761	3034	10066	10	79	Teemo	1	5423	5926	4277	11	109	1
16	Sejuani	1	5055	4190	4307	11	106	Sett	0	4153	2705	6767	10	111	1
17	Vayne	5	6577	11088	9160	11	81	Fiora	4	5543	5990	12896	9	66	0
18	Darius	4	7842	9096	10562	12	121	Volibear	0	3445	5615	9285	9	60	0
19	Ornn	1	4640	7972	11102	11	103	Rengar	2	4897	9426	9641	11	104	1
20	Rengar	3	6490	6664	6279	11	122	Akali	1	4016	3374	5498	9	74	1
21	Aatrox	5	7757	6908	7097	11	100	Olaf	0	3061	4359	8180	7	51	0

그림 14 : topdata 테이블 데이터

데이터 전처리

1. 데이터 준비

- 데이터 전처리를 진행할 테이블을 DataFrame으로 변환
- data type 확인
- 챔피언이 문자열로 저장되어 있어 riot에서 제공해주는 고유 정수로 변환

```
con = pymysql.connect(host="127.0.0.1", db="riot", user="root", password="root1234", charset='utf8')
topdata = pd.read_sql("select * from topdata", con)
topdata
```

```
D:₩kdigital₩anaconda₩lib₩site-packages₩pandas₩io₩sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlit
onsider using SQLAlchemy
  warnings.warn(
```

	tno	blue_top_champ	bt_kill	bt_gold	bt_damageDone	bt_damageTaken	bt_level	bt_minion	red_top_champ	rt_kill	rt_gold	rt_damageDone	rt_damageTaken	rt_level	rt_minion	result
0	1	Gangplank	1	5256	11067	12651	10	93	Fiora	3	4944	11610	15583	11	90	1
1	2	Zac	3	5987	9122	12814	11	114	Renekton	0	5342	8326	13051	10	125	0
2	3	Akali	1	5486	4786	5463	11	129	Fiora	0	4259	3132	6348	10	112	0
3	4	Aatrox	3	5633	6102	9976	10	91	Jax	1	4087	5130	11299	9	50	1
4	5	Sejuani	4	6880	7338	9899	11	119	Nocturne	1	5083	5966	8925	10	84	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29543	29545	Gwen	1	4361	7908	10619	9	90	Volibear	4	6525	9044	11379	11	122	1
29544	29546	Aatrox	2	5799	5159	7760	11	128	Shyvana	0	4241	4455	6399	11	102	0
29545	29547	Nasus	0	3848	3474	10524	9	83	Gwen	2	6460	8118	8260	11	124	1

그림 15 : topdata 테이블 DataFrame으로 변환

```
# 데이터 타입 확인
topdata = topdata.drop(['tno'],axis=1)
topdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29548 entries, 0 to 29547
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   blue_top_champ   29548 non-null  object
 1   bt_kill          29548 non-null  int64
 2   bt_gold          29548 non-null  int64
 3   bt_damageDone    29548 non-null  int64
 4   bt_damageTaken   29548 non-null  int64
 5   bt_level         29548 non-null  int64
 6   bt_minion        29548 non-null  int64
 7   red_top_champ    29548 non-null  object
 8   rt_kill          29548 non-null  int64
 9   rt_gold          29548 non-null  int64
 10  rt_damageDone    29548 non-null  int64
 11  rt_damageTaken   29548 non-null  int64
 12  rt_level         29548 non-null  int64
 13  rt_minion        29548 non-null  int64
 14  result           29548 non-null  int64
dtypes: int64(13), object(2)
memory usage: 3.4+ MB
```

```
# champ code 불러오기
with open('champ.pickle','rb') as f:
    champCode = pickle.load(f)

champCode[['champion_key','en_name']]
```

	champion_key	en_name
0	266	Aatrox
1	103	Ahri
2	84	Akali
3	166	Akshan
4	12	Alistar
...	...	...
156	221	Zeri
157	115	Ziggs
158	26	Zilean
159	142	Zoe
160	143	Zyra

그림 16 : topdata 정보확인 및 챔피언 라벨링에 활용할 데이터 피클 로드

```
# 챔피언명 라벨링
def changeCode(x):
    for code, name in champCode[['champion_key','en_name']].iterrows():
        if x == name[1]:
            return name[0]
red_top_champ_code = topdata['red_top_champ'].apply(changeCode)
blue_top_champ_code = topdata['blue_top_champ'].apply(changeCode)

chmapToCode = pd.concat([red_top_champ_code,blue_top_champ_code],axis=1)
chmapToCode
```

	red_top_champ	blue_top_champ
0	114	41
1	58	154
2	114	84
3	24	266
4	56	113
...	...	...
29543	106	887
29544	102	266
29545	887	75
29546	41	150
29547	22	69

그림 17 : blue_top_champ, red_top_champ 두 컬럼 라벨링

데이터 전처리

2. 데이터 가공

- 라밸링된 컬럼을 topdata데이터와 결합
- 결측치 제거
- object타입을 int로 변환
- 플레이 횟수가 15 미만인 챔피언 데이터를 삭제
- 인덱스 재설정

```
#라벨링된 데이터로변경
labelTop = pd.concat([chmapToCode,topdata.drop(['blue_top_champ','red_top_champ'], axis=1)],axis=1)

#결측치 제거
labelTop = labelTop.dropna(axis=0)

#타입변경
labelTop = labelTop.astype({'red_top_champ':'int64'})
labelTop = labelTop.astype({'blue_top_champ':'int64'})
labelTop.info()
labelTop
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29322 entries, 0 to 29547
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   red_top_champ   29322 non-null  int64
 1   blue_top_champ  29322 non-null  int64
 2   bt_kill         29322 non-null  int64
 3   bt_gold         29322 non-null  int64
 4   bt_damageDone   29322 non-null  int64
 5   bt_damageTaken  29322 non-null  int64
 6   bt_level        29322 non-null  int64
 7   bt_minion       29322 non-null  int64
 8   rt_kill         29322 non-null  int64
 9   rt_gold         29322 non-null  int64
 10  rt_damageDone   29322 non-null  int64
 11  rt_damageTaken  29322 non-null  int64
 12  rt_level        29322 non-null  int64
 13  rt_minion       29322 non-null  int64
 14  result          29322 non-null  int64
dtypes: int64(15)
```

그림 18 : 라밸링 > 결측치제거 > int로 변환 코드

```
# 비주류챔프 삭제
df_red = pd.DataFrame(topdata.red_top_champ.value_counts() < 15).reset_index()
df_blue = pd.DataFrame(topdata.blue_top_champ.value_counts() < 15).reset_index()
# df_red.loc[df_red['red_top_champ'] == True]['index'],df_blue.loc[df_blue['blue_top_champ'] == True]['index']
delList_red = list(df_red.loc[df_red['red_top_champ'] == True]['index'])
delList_blue = list(df_blue.loc[df_blue['blue_top_champ'] == True]['index'])
target_red = []
target_blue = []
for i, row in topdata.iterrows():
    if row['red_top_champ'] in delList_red:
        print(i, row['red_top_champ'])
        target_red.append(i)
for i, row in topdata.iterrows():
    if row['blue_top_champ'] in delList_blue:
        print(i, row['blue_top_champ'])
        target_blue.append(i)
    #if v in delList:
topdata = topdata.drop(index=target_red)
topdata = topdata.drop(index=target_blue)
topdata
```

```
172 427
292 40
```

그림 19 : 비주류 챔프 삭제 코드

데이터 분석

1. 데이터 분석

- 사용할 분석기법 선택 (로지스틱)
- 학습용, 시험용 데이터 분할
- 정확도, 정밀도, 재현율 측정

데이터분석

```
In [20]:
from sklearn.model_selection import train_test_split

In [379]:
# 데이터 분할
x_train, x_test, y_train, y_test=train_test_split(topdata.drop('result',axis=1), topdata['result'], test_size=0.21)
len(x_train), len(x_test), len(y_train), len(y_test)

Out[379]: (22980, 6109, 22980, 6109)

In [394]:
# 로지스틱 회귀분석으로 모델학습
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(solver='liblinear', random_state=1, C=0.01, )
model.fit(x_train, y_train)
pred = model.predict(x_test)
pred

Out[394]: array([1, 0, 1, ..., 0, 1, 1], dtype=int64)

In [395]:
from sklearn.metrics import accuracy_score, precision_score, recall_score
# 모델의 정확도,정밀도,재현율
accuracy_score(pred,y_test),precision_score(pred,y_test),recall_score(pred,y_test)

Out[395]: (0.6460959240464887, 0.6349822179114128, 0.655321988655322)
```

그림 20 : 분석과정 및 결과 화면

2. 분석 결과

- **Accuracy(정확도) : 약 65%**
- **Precision(정밀도) : 약 64%**
- **Recall(재현율) : 약 65.5%**

3. 마무리

시간이 부족하여 목표로 했던 모든 포지션을 분석하지는 못하였으나 추후 지속하여 모든 포지션의 분석을 완료하려고 한다. 2가지 이유로 원하는 결과를 얻지 못한 것 같다. 첫째로는 분석을 위해 선정된 데이터 컬럼의 수가 부족한 것 같고 둘째로는 League Of Legend라는 게임이 하나의 포지션 양성에 따라 게임 결과에 모든 것을 기여하는 것은 아니기에 어떻게 보면 65%라는 수치가 낮지 않은 수치라고 생각된다.

프로젝트 업데이트 2022.12.06

업데이트 사항 : jungle, mid, AD 포지션 분석 추가

1. 분석 과정

상기 TOP포지션 분석 과정과 동일

2. 분석 결과

2-1. Jungle

- **Accuracy(정확도) : 약 68%**
- **Precision(정밀도) : 약 66%**
- **Recall(재현율) : 약 68%**

2-2. Mid

- **Accuracy(정확도) : 약 68%**
- **Precision(정밀도) : 약 67%**
- **Recall(재현율) : 약 69%**

2-3. AD

- **Accuracy(정확도) : 약 69%**
- **Precision(정밀도) : 약 70%**
- **Recall(재현율) : 약 68%**

3. 마무리

모든 포지션 분석결과 상기 TOP라인의 분석결과보다 좋은 결과를 도출하였습니다. 현재 LOL이라는 게임이 AD 및 MID가 중요시되는 메타로 정확도가 70내외로 높은 편은 아니나, 현 메타를 반영하고 있어 방향성은 맞았다고 판단이 됩니다. 그러나 더 높은 정확도를 위해 RIOT API에서 제공하는 데이터 컬럼을 늘려 좀 더 세밀하게 분석을 진행해보려고 합니다.