

2021-2 YBIGTA 컨퍼런스

# GameBoard



배지호, 안범진, 오창준, 유동준, 이종우, 장수용

# GameBoard

---





게임.. 다들 좋아하시죠?  
저희도 좋아해요..  
그래서 해봤습니다

# GameBoard


## “GameBoard?”

### game

미국·영국[ɡeɪm]  영국식 


#### 명사

1 게임, 경기, 시합 (→war **game**)

card games 

카드 게임


2 게임[경기/시합]하기

to play a **game** of chess 

체스 게임을 하다[체스를 한 판 두다]


#### 형용사

1 투지만만한

She's **game** for anything. 


그녀는 무엇이든 할 투지를 갖고 있다.

### board

미국식[bɔ:rd]  영국식[bɔ:d] 

#### 명사

1 판자, 널 (→chipboard, floorboard, hardboard, skirting **board**)

He had ripped up the carpet, leaving only the bare boards. 

그가 카펫을 걷어 내고 맨 마룻장만 남겨 놓은 상태였다.


2 (판자같이 생긴) -판[-대] (→message **board**)

a blackboard 

칠판


#### 동사

1 승선[승차/탑승]하다

The ship was boarded by customs officials. 

그 배에는 세관 직원들이 승선에 있었다.

2 (비행기·배가) 탑승[승선]에 들어가다

Flight BA193 for Paris is now boarding at Gate 37. 

파리행 BA193 여객기가 지금 37번 게이트에서 탑승을 실시하고 있습니다.

투지만만한 판자..!

# GameBoard

---

“GameBoard?”

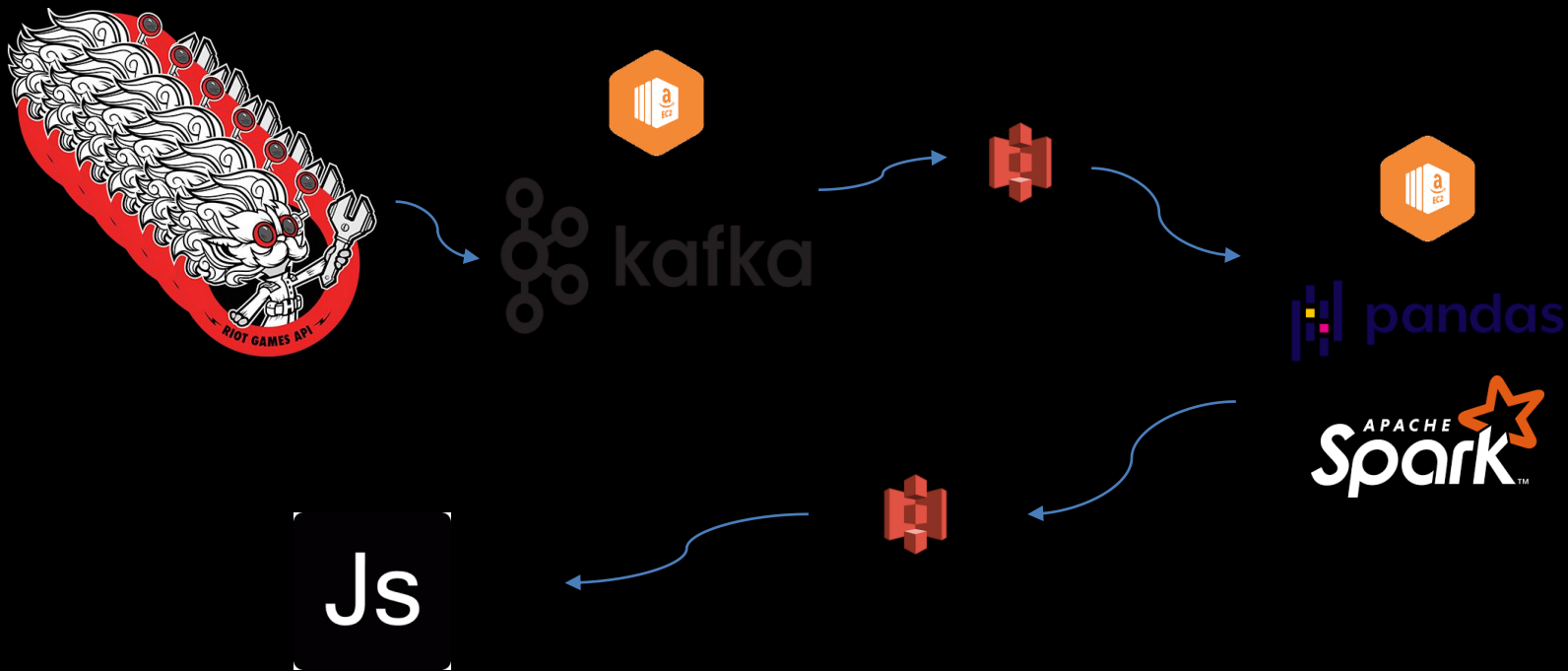
게임 대시보드

리그오브레전드의 각종 지표 분석

지표를 활용한 통계, 및 시각화

정확.. 할 거예요

# Architecture



# LOL API / Producer

<https://developer.riotgames.com/>



## ACCOUNT-V1

Accounts RSO

## CHAMPION-MASTERY-V4

League of Legends

## CHAMPION-V3

League of Legends

## CLASH-V1

League of Legends

## LEAGUE-EXP-V4

League of Legends

## LEAGUE-V4

League of Legends

## LOL-STATUS-V3

League of Legends

## LOL-STATUS-V4

Teamfight Tactics

## LOR-DECK-V1

Legends of Runeterra RSO

## LOR-INVENTORY-V1

Legends of Runeterra RSO

```
#특정 유저의 닉네임으로 puuid 조회
def get_puuid_by_id(user_id):
    url = 'https://asia.api.riotgames.com/riot/account/v1/accounts/by-riot-id/'
    tag = 'XZ'
    header = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
        "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": token
    }

    res = requests.get(url+user_id+"/"+tag, headers = header)
    if res.status_code != 200:
        print(res.text)
        print('get puuid by id', user_id)
        print("ERROR OCCURED", res.status_code)
        puuid = json.loads(res.text)['puuid']
        return (puuid, res.status_code)

#유저 아이디를 상세정보 받아옴
def get_detail_user_from_id(user_id):
    url = 'https://kr.api.riotgames.com/lo/summoner/v4/summoners/by-name/' + user_id
    header = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
        "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": token
    }

    res = requests.get(url, headers = header)
    if res.status_code != 200:
        print(res.text)
        print('get detail user from id', user_id)
        print("ERROR OCCURED", res.status_code)
        user_detail = json.loads(res.text)
        return (user_detail, res.status_code)

for tier in tier_list:
    for division in division_list:
        for page in range(1, 100000):
            print("PAGE :", page)
            now_user_list = get_user_list_from_tier(tier, division, page)
            apis += 1
            if len(now_user_list[0]) == 0:
                break #페이지 끝났음 따 여기와.
            for user in now_user_list[0]:
                print(user)
                try:
                    name = user['summonerName']
                    puuid = get_puuid_by_id(name)[0]
                    game_list = get_matches_from_puuid(puuid, 10)
                except Exception as e:
                    print(str(e) + " " + user['summonerName'] + '\n')
                    log(str(e) + " " + user['summonerName'] + '\n')
                    continue #puuid 없는 아이디는 넘어가자.
                time.sleep(2)
                apis += 1
            for game in game_list[0]:
                now_game = get_match_detail_from_game_id(game) #여기선 여러 나는 부분이 없다.
                if now_game[1] == 200: #유용한 응답
                    real_game = now_game[0]
                    if real_game['info']['gameVersion'][:5] == now_version: #ok
                        producer.send("real_"+tier+"_"+division, value = json.dumps(real_game)).get()
                        print("SUCCESS")
                    else: #버전 기반 게임종류
                        break
                apis += 1
                time.sleep(2)
            print(apis)
```

# LOL API / Producer

<https://developer.riotgames.com/>



```
{
  "result": {
    {
      "metadata": {
        "dataVersion": "2",
        "matchId": "KR_5632000336",
        "participants": [
          "501cbbauHs6dKgUj2EZUBMTAPk7QHUVqJUCNFXSIEPGPGpZ-4vm96iLoQu1Fcuw-Jjrcwib9ecQ",
          "Jp0Agix74f12q45W24xAln9iaYgyWdKtFFgVzK6AqGh8dWfAI2T2Z0bX6320cn3d1At-KOA4IQ",
          "ERx_95aDBNORJ5Z3_Sel-LrUKW5Z-1F4UXHtBGOZC7z7N-5hqYbYDyeh1YymxhXc-XqAZHFENzQ",
          "VXMp48glChVsfQpkOxethSgW0h69c-Q4dhZER0OHJR3q1Xeq3LhndV5F1KP8MTbe4o-AC8Pghw",
          "XlgwQt1P40CFZQd-ie3i00WGKy7TMgT6OEW_VOWuAuojs9LerCfOfaj26xdfutRWOd5UAkQ",
          "9y87YQuVQGKIS_Ni3dJUMU6cwtWaaE8e0HTPp9VcIPBHzNw3IP7051cJLpAQNH9MZgmL62ehfJgQA",
          "JyK2jaeBvLr1t9avyQifMA5rOqKge5rWbQX8Vv4D116LtgjnbSP1TNFK9CnhNyzhv87G_DYxg",
          "_2Q_siFpV-gjieSQ9NuEsJlvfOoD4OuiqVGXx26VqwkXJmgF2mwfpulSV-91NngyHSBzELeiw",
          "cUzqKUttfCpXLqrSVB9XvcfAA5_PmM3Ot1WJ6YjgKK7hXWHbYzPzi-tmz1vEccs-rfM5AmoA",
          "drK_2oelOokksrfjw9S5n5zgI3Hhuzz84aUfJdUvJedh19h4nZFXnNf48gpXb8CDxxz-gwYfjw"
        ]
      }
    }
  },
  "info": {
    "gameCreation": 1639740683000,
    "gameDuration": 1810,
    "gameEndTimestamp": 1639742527122,
    "gameId": 5632000336,
    "gameMode": "CLASSIC",
    "gameName": "teambuilder-match-5632000336",
    "gameStartTimestamp": 1639740716304,
    "gameType": "MATCHED_GAME",
    "gameVersion": "11.24.413.2485",
    "mapId": 11,
    "participants": [
      {
        "assists": 10,
        "baronKills": 1,
        "bountyLevel": 0,
        "champExperience": 14924,
        "champLevel": 16,
        "championId": 876,
        "championName": "Lillia",
        "championTransform": 0,
        "consumablesPurchased": 4,
        "damageDealtToBuildings": 1698,
        "damageDealtToObjectives": 26012,
        "damageDealtToTurrets": 1698,
        "damageSelfInflicted": 14633,
        "deaths": 4,

```

```
gameCreation = 1637923439000 #게임 만들어진 unix 시간인듯
gameDuration = 2511 #1150초 동안 게임 진행
gameEndTimestamp = 1637925981489 #게임 끝난 unix 시간인듯 (Creation과 2542.489도 gameDuration
gameId = 5590242329 #KR_5590242329
gameMode = 'CLASSIC' #이게 한국이고 ARAM이 할바람인듯
gameName = 'teambuilder-match-5590242329'
gameStartTimestamp = 1637923469096 #end - start = duration, Creation은 로딩중부터 인듯
gameType = 'WATCHED_GAME' #랭크, 일반전 나누는것 같은데
gameVersion = '11.23.409.111'
mapId = 11
participants = 팀원마다의 상세 정보11 밑에서 다들
platformId = 'KR'
queueId = 420
teams = [{"bans": [{"championId": 22, "pickTurn": 1},
!!!participants!!!
앞에서 10만 list. 유제마디 dict 하나씩.
gameInfo['info']["participants"].keys() -> ['assists', 'baronKills', 'bountyLevel',
'champExperience', 'champLevel', 'championId', 'championName', 'championTransform',
'consumablesPurchased', 'damageDealtToBuildings', 'damageDealtToObjectives',
'damageDealtToTurrets', 'damageSelfInflicted', 'deaths', 'detectorWardsPlaced',
'doubleKills', 'dragonKills', 'firstBloodAssist', 'firstBloodKill', 'firstTowerAssist',
'firstTowerKill', 'gameEndedInEarlySurrender', 'gameEndedInSurrender', 'goldEarned',
goldSpent', 'individualPosition', 'inhibitorKills', 'inhibitorTakedowns',
'inhibitorLost', 'item0', 'item1', 'item2', 'item3', 'item4', 'item5', 'item6',
'itemsPurchased', 'killingsPrees', 'kills', 'lane', 'largestCriticalStrike',
'largestKillingSpree', 'largestMultiKill', 'longestTimeSpentLiving', 'magicDamageDealt',
'magicDamageDealtToChampions', 'magicDamageTaken', 'neutralMinionsKilled', 'nexusKills',
'nexusLost', 'nexusTakedowns', 'objectivesStolen', 'objectivesStolenAssists',
'participantId', 'pentaKills', 'perks', 'physicalDamageDealt',
'physicalDamageDealtToChampions', 'physicalDamageTaken', 'profileIcon', 'puuid',
'quadrakills', 'riotIdName', 'riotIdFlagline', 'role', 'sightWardsBoughtInGame',
'spellCasts', 'spell2Casts', 'spell3Casts', 'spell4Casts', 'summonerCasts',
'summonerId', 'summonerKills', 'summonerLevel', 'summonerName', 'summonerPoints',
'summonerType', 'teamEarlySurrendered', 'teamId', 'teamPosition', 'timeClingOthers',
'timePlayed', 'totalDamageDealt', 'totalDamageDealtToChampions', 'totalDamageDealtToTeamates', 'totalDamageTaken', 'totalHeal',
'totalHealsOnTeamates', 'totalMinionsKilled', 'totalTimeCDealt', 'totalTimeSpentDead',
'totalUnitsHealed', 'tripleKills', 'trueDamageDealt', 'trueDamageDealtToChampions',
'trueDamageTaken', 'turretKills', 'turretTakedowns', 'turretsLost', 'unrealKills',
'visionScore', 'visionWardsBoughtInGame', 'wardsKilled', 'wardsPlaced', 'win']
```

# LOL API / Producer

<https://developer.riotgames.com/>



This API key is to be used for development only. Please register any permanent products.

**Do NOT use this API key in a publicly available product!**

.....

Show

Copy

**Expired: Wed, Dec 22nd, 2021 @ 11:38pm (PT)**

Your key has expired. You must regenerate your API key.

## RATE LIMITS

20 requests every 1 seconds(s)

100 requests every 2 minutes(s)

Note that rate limits are enforced per routing value (e.g., na1, euw1, americas).

1초에 20번  
2분에 100번

하루 = 72,000번 요청 가능



# LOL API / Producer

<https://developer.riotgames.com/>



jh	asia-east1-b
jw	asia-east1-b
kafka-broker-2	asia-east1-b
kafka-producer-1	asia-east1-b
kafka-producer-2	asia-east1-b
kafka-producer-3	asia-east1-b
kafka-test-1	asia-east1-b
spark-1	asia-northeast3-a
sy	asia-east1-b

kafka-producer1,2,3  
jh, jw, sy

Diamond – I, II, III  
Platinum – I, II, III

Send to Kafka  
Kafka-python  
Json

```
from kafka import KafkaProducer
from json import dumps
import time
producer = KafkaProducer(acks=0, compression_type='gzip',
    bootstrap_servers=['35.189.189.115:9092'],
    value_serializer=lambda x: dumps(x).encode('utf-8'))
```

# Kafka / EC2 / Consumer

---

<https://kafka.apache.org/>



- 데이터 스트리밍을 위해 데이터를 저장해 놓을 수 있는 일종의 큐
- Topology – Topology 연결로 인한 아키텍처 복잡성을 극복하고자 개발됨

EC2  
Zookeeper 3.4.10  
Kafka 2.11

6개의 Producer에서 받은 데이터를  
real\_DIAMOND\_I  
real\_DIAMOND\_II  
real\_DIAMOND\_III  
real\_PLATINUM\_I  
real\_PLATINUM\_II  
real\_PLATINUM\_III

토픽에 저장

Consumer

카프카에 저장된 데이터를 S3에 저장하는 역할  
하나의 EC2에서 멀티쓰레딩으로  
6개의 토픽 한번에 관리

# Pandas & SPARK

<https://pandas.pydata.org/> , <https://spark.apache.org/>



- Pandas : 다양한 언어로 사용할 수 있는 데이터 분석 패키지
- Spark : 분산처리를 통한 데이터 분석 프레임워크 (3.2.0)
- Pandas와 Spark에서 동시에 데이터 분석 진행 – 더 잘되는 쪽으로..!
- 그리고 Pandas에서 오래 걸리는 작업을 Spark로 진행
- Json → DataFrame 으로 변형해 분석 진행

1. 승률, 픽률, 밴률 분석
2. 라인별 픽률 분석
3. 승리를 결정하는 요인 분석
4. 이번 시즌 트렌드 분석

분석한 결과를 다시 Json 형태로 S3에 저장!

```
bans = ban.withColumn("ban_detail", explode(col("ban"))).drop("ban")
baron = bans.select("ban_detail.objectives.baron.first").withColumnRenamed("first", "baron")
champion = bans.select("ban_detail.objectives.champion.first").withColumnRenamed("first", "champion")
dragon = bans.select("ban_detail.objectives.dragon.first").withColumnRenamed("first", "dragon")
inhibitor = bans.select("ban_detail.objectives.inhibitor.first").withColumnRenamed("first", "inhibitor")
riftHerald = bans.select("ban_detail.objectives.riftHerald.first").withColumnRenamed("first", "riftHerald")
tower = bans.select("ban_detail.objectives.tower.first").withColumnRenamed("first", "tower")
win = bans.select("ban_detail.win")

#column concatenation
schema = StructType(baron.schema.fields + champion.schema.fields)
res = baron.rdd.zip(champion.rdd).map(lambda x : x[0] + x[1])
DF = spark.createDataFrame(res, schema)

schema = StructType(DF.schema.fields + dragon.schema.fields)
res = DF.rdd.zip(dragon.rdd).map(lambda x : x[0] + x[1])
DF = spark.createDataFrame(res, schema)

schema = StructType(DF.schema.fields + inhibitor.schema.fields)
res = DF.rdd.zip(inhibitor.rdd).map(lambda x : x[0] + x[1])
DF = spark.createDataFrame(res, schema)

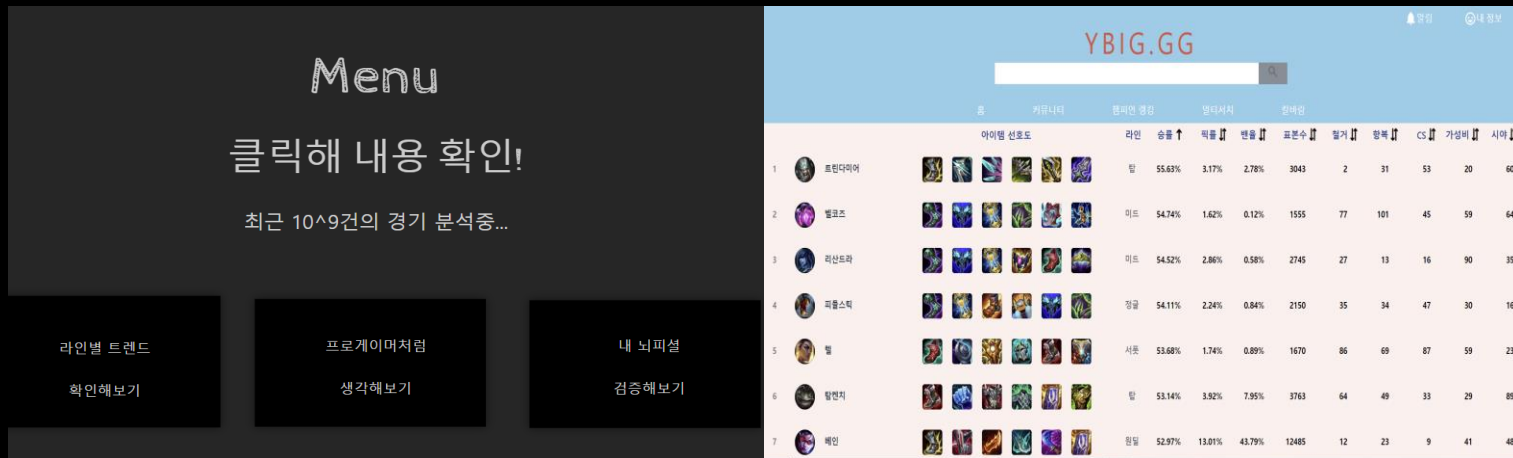
schema = StructType(DF.schema.fields + riftHerald.schema.fields)
res = DF.rdd.zip(riftHerald.rdd).map(lambda x : x[0] + x[1])
DF = spark.createDataFrame(res, schema)

schema = StructType(DF.schema.fields + tower.schema.fields)
res = DF.rdd.zip(tower.rdd).map(lambda x : x[0] + x[1])
DF = spark.createDataFrame(res, schema)

schema = StructType(DF.schema.fields + win.schema.fields)
res = DF.rdd.zip(win.rdd).map(lambda x : x[0] + x[1])
DF = spark.createDataFrame(res, schema)
```

# Js

- <https://lutris98.github.io/GameBoard/#/1>, <https://github.com/Lutris98/GameBoard/tree/main/lib>**



# APIs

---

- 사용한 유저 티어 – platinum 1,2,3 diamond 1,2,3
- 가장 먼저 각 티어에 존재하는 유저 리스트를 받음 → 아래와 같은 json이 205개 들어있는 리스트 받음
- 각 티어별로 205명씩 받아서 모든 유저 데이터를 추출

```
{
  'leagueId': '4fbe881f-209d-447a-9196-291a09530115',
  'queueType': 'RANKED_SOLO_5x5',
  'tier': 'PLATINUM',
  'rank': 'I',
  'summonerId': 'zxUQgv0MoHWrm dB eQLyGGw5Y98AMay3eR5vKE2m10Q8GDxc_',
  'summonerName': 'Apsiod',
  'leaguePoints': 75,
  'wins': 282,
  'losses': 308,
  'veteran': False,
  'inactive': False,
  'freshBlood': False,
  'hotStreak': False},
```

# APIs

---

- 추출한 유저 데이터에서 해시된 id를 나타내는 puuid를 추출
- 그리고 유저의 최근 10경기의 matchId를 받아 옴

```
[ 'KR_5647344404',  
  'KR_5647146984',  
  'KR_5647070011',  
  'KR_5647040529',  
  'KR_5644412659',  
  'KR_5644296005',  
  'KR_5640941957',  
  'KR_5640834514',  
  'KR_5640801373',  
  'KR_5640763449' ]
```













# APIs

- **matchId**를 이용해 각 게임의 상세 데이터 조회
- 이 데이터를 100개씩 묶어 json으로 바꾼 후 S3에 저장!

```

{
  "metadata": {
    "dataVersion": "2",
    "matchId": "KR_564734404"
  },
  "participants": [
    {
      "participantId": 1,
      "teamId": 1,
      "championId": "Yasuo",
      "summonerId": "KR_564734404",
      "gameVersion": "11.24.414.4003",
      "mapId": 11,
      "participants": [
        {
          "assists": 6,
          "baronKills": 0,
          "bountyLevel": 0,
          "champExperience": 13913,
          "champLevel": 15,
          "championId": 157,
          "championName": "Yasuo",
          "championTransform": 0,
          "consumablesPurchased": 11,
          "damageDealtToBuildings": 2908,
          "damageDealtToObjectives": 2908,
          "damageDealtToTurrets": 2908,
          "damageSelfMitigated": 28267,
          "deaths": 20,
          "detectorWardsPlaced": 4,

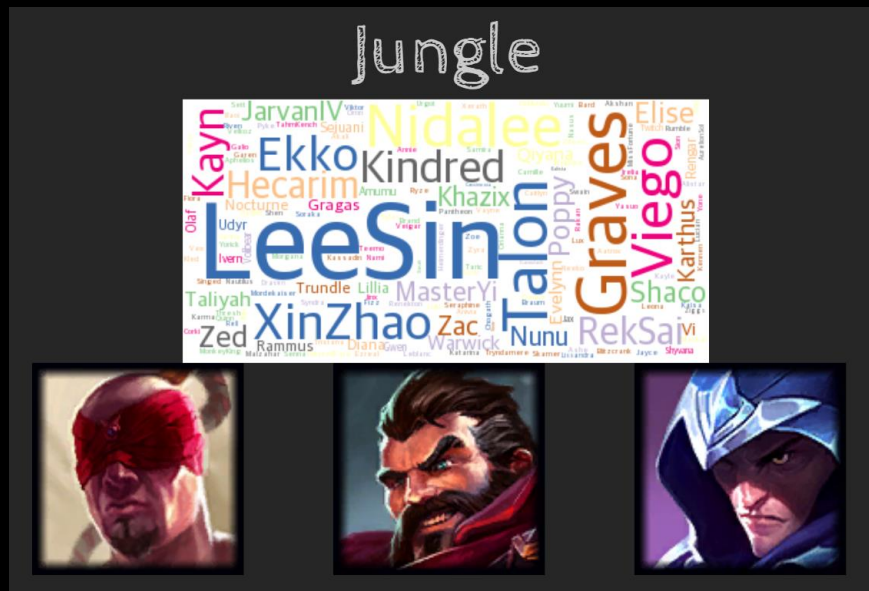
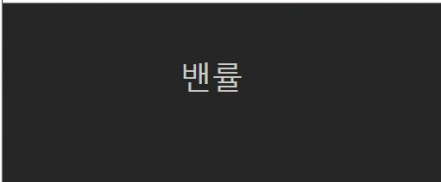
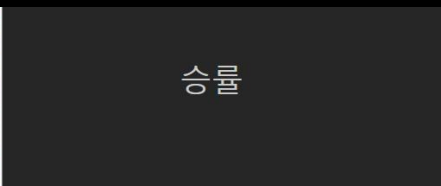
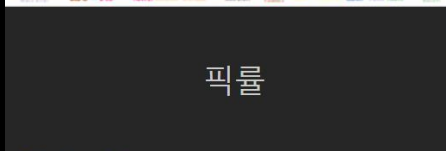
```

 <a href="#">real_DIAMOND_III-20211222153255.json</a>	json	2021. 12. 23. am 12:32:56 AM KST	7.9MB
 <a href="#">real_DIAMOND_III-20211222153724.json</a>	json	2021. 12. 23. am 12:37:26 AM KST	7.9MB
 <a href="#">real_PLATINUM_I-20211222122947.json</a>	json	2021. 12. 22. pm 9:29:51 PM KST	8.0MB
 <a href="#">real_PLATINUM_I-20211222122951.json</a>	json	2021. 12. 22. pm 9:29:54 PM KST	7.9MB
 <a href="#">real_PLATINUM_I-20211222123142.json</a>	json	2021. 12. 22. pm 9:31:45 PM KST	8.0MB
 <a href="#">real_PLATINUM_I-20211222123146.json</a>	json	2021. 12. 22. pm 9:31:49 PM KST	7.9MB
 <a href="#">real_PLATINUM_I-20211222123149.json</a>	json	2021. 12. 22. pm 9:31:52 PM KST	7.9MB
 <a href="#">real_PLATINUM_I-20211222123152.json</a>	json	2021. 12. 22. pm 9:31:55 PM KST	7.9MB
 <a href="#">real_PLATINUM_I-20211222123155.json</a>	json	2021. 12. 22. pm 9:31:56 PM KST	7.9MB
 <a href="#">real_PLATINUM_I-20211222123157.json</a>	json	2021. 12. 22. pm 9:31:58 PM KST	7.9MB
 <a href="#">real_PLATINUM_I-20211222123158.json</a>	json	2021. 12. 22. pm 9:32:01 PM KST	7.9MB
 <a href="#">real_PLATINUM_I-20211222123201.json</a>	json	2021. 12. 22. pm 9:32:03 PM KST	7.9MB

# Analysis / Visualization

## 1. 라인별 트렌드 확인

챔피언의 승률, 픽률, 밴률을 분석 → Tag Cloud 구성

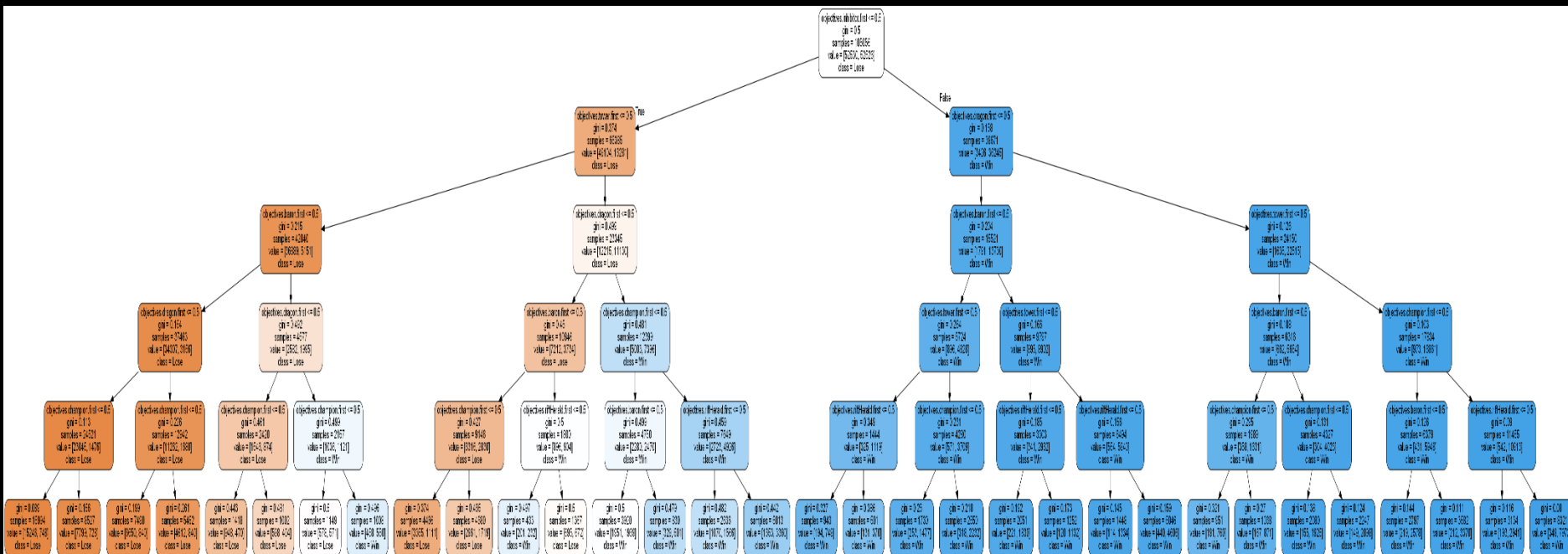




# Analysis / Visualization

## 2. 승리를 결정짓는 요소 분석

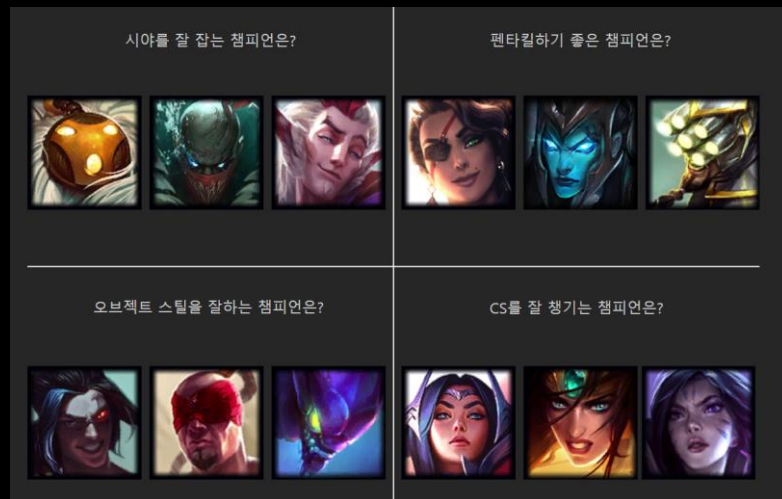
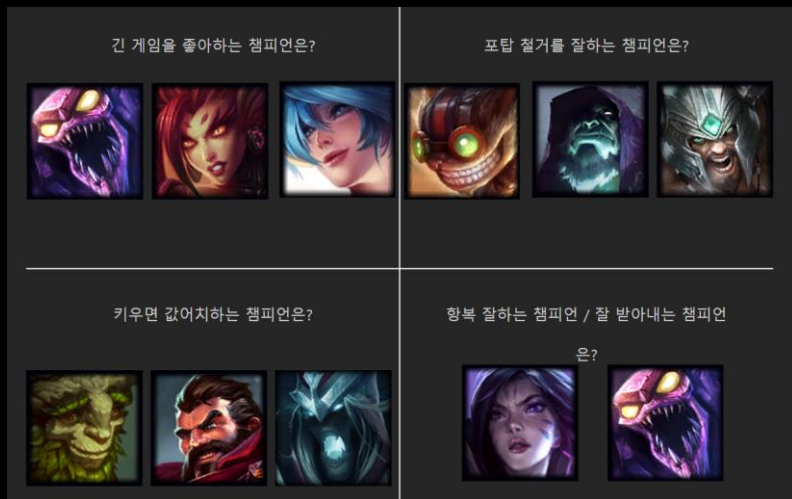
첫 킬, 첫 드래곤, 첫 타워 등의 True/False 데이터를 가지고 Decision Tree 적용



# Analysis / Visualization

## 3. 독특한 요소 추출

평균 게임시간이 긴 챔피언들, 포탑을 많이 부신 챔피언들, 시야를 잘 잡는 챔피언들 등등 ..



**Thank you**

---