**졸업프로젝트 보고서**

**게임 데이터 분석 및 시각화**

**(League of Legends)**

2013171046 심재훈

Data and Visual Analytics Lab

고려대학교 정보대학 컴퓨터학과

**1. Introduction**

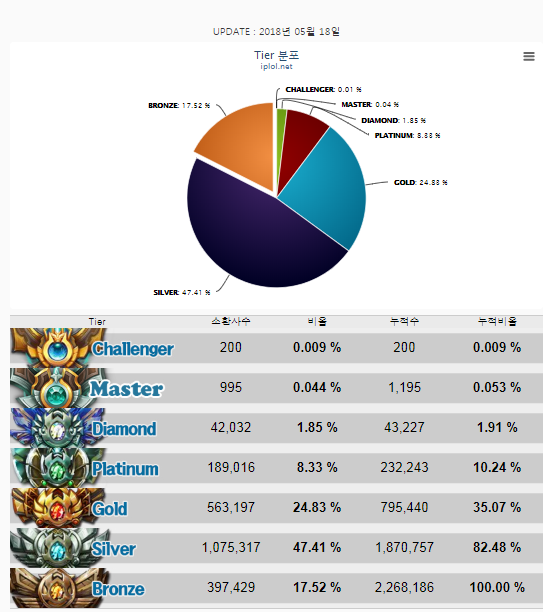
* **Problem Statement**
  + League of Legends(이하롤)에서 나오는 게임 데이터로부터 승리에 핵심적인 요인을 분석해 보려고 한다. 분석한 결과의 시각화를 통해 직관적으로 어느 전략이 좋을지 표현해 보려고 한다. 단순히 기술 통계를 이용한 분석보다는 선형 회귀선이나 머신러닝 알고리즘을 이용해 승리에 핵심적인 요인을 분석해보려고 한다.
* **Pains and Needs** 
  + 데이터가 방대한 량이기 때문에 api를 이용해 특정 데이터만 추출해낼 수 있다. 추출된 데이터는 json 형식으로 nosql type이다. 이를 table 형식으로 가공해야만 데이터 분석을 원활히 시행할 수 있다. 또한 야구에서의 포지션처럼 롤에서도 포지션이 존재하는데 이에 따라 구분 지어 데이터 분석을 하게 되면 데이터가 sparse 해질 수 있다. 어떤 데이터를 쓸지에 따라 알고리즘의 성능이 많이 바뀌기 때문에 sparse한 데이터의 문제점을 처리해야만 한다.
* **Importance**
  + 5 : 5 대전인 온라인 게임 데이터 분석은 야구 데이터 분석과 유사한 점이 많다. 온라인 특성상 야구 데이터보다 객관적이고 풍부한 데이터를 얻을 수 있는데 이를 다루면 전략을 분석해내는 알고리즘을 더 발전시킬 수 있을 것이다.

**2. Related Work (Project)**

* **풀고자하는연구내용(Problem)을다른논문/프로젝트들의저자들은어떻게풀고있는가? 해당문제를풀기위해서기존에는어떤접근법(approach)들이존재했는가? 그것들의장단점은무엇인가?**

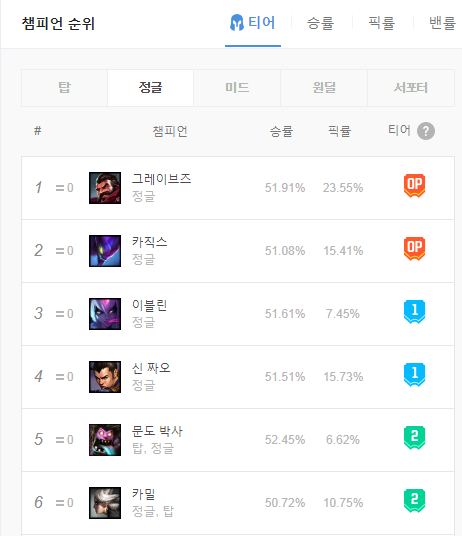
롤의 기본 정보: 롤에서는 2018년 5월기준 140명의 챔피언들이 있고 각 팀 내의 5명은 이들 중 각각의 역할군에 맞는 챔피언을 골라(중복x) 상대한다. 각 역할군은 TOP, MID, BOTTOM, JUNGLE이 있고 BOTTOM의 경우 보통 2명이 당담한다. 목표는 상대팀 진영의 넥서스(최종 건물)을 부수는 것이다. 순위를 매기는 랭크게임의 경우 각 팀원마다 한가지 챔피언을 밴(ban)할 수 있고(중복o) 밴이 되면 그 챔피언은 해당 게임에서 이용 하지 못 한다. 랭크의 경우 수준별 리그로 묶는데 이 리그들로는 잘하는 순서대로 챌린저, 마스터, 다이아, 플래티넘, 골드, 실버, 브론즈가 있다(pg6 그림 1 첨부).게임 내에서는 목표물을 잡아 돈을 벌 수 있고 돈을 이용해 게임 내 영향력을 주는 아이템을 구매할 수 있다.

롤의 데이터분석 사이트로 fow.kr, op.gg, your.gg 3가지가 있다.

fow.kr 에서는 챔피언통계를 제공한다. 챔피언의 픽률(게임에서 선택되는 비율), 밴율, 승률을 보여주고 아이템의 선택되는 비율을 보여준다. 각 리그마다의 기술통계를 볼 수 있어 상위권 리그에서 어느 챔피언이 좋게 쓰이는지, 어떤 아이템을 선택하는 것이 좋을지 참고하는데 유용하다. 이러한 기술통계적 분석은 다른 사이트에서도 기본적으로 제시되어 있다. 단순히 수치들의 평균값만 나열되어 있기 때문에 실제로 무엇이 유용한지 찾으려면 시간이 많이 필요하다. 가령 어느 한 챔피언의 픽률이 높다고 해서 그 챔피언의 승률이 낮다면 해당 챔피언을 쓰는 것에 대해 고민해볼 필요가 있다.

출처: <http://iplol.co.kr/leaguedist.do>

**<그림1 롤티어분포도>**



* 티어행을보면챔피언별등급을확일할수있다. OP, 1, 2,...순으로좋다.
* 출처: http://www.op.gg/champion/statistics

**<그림2 op.gg챔피언별등급>**



출처: https://your.gg/kr/

**<그림3 your.gg그래프>**

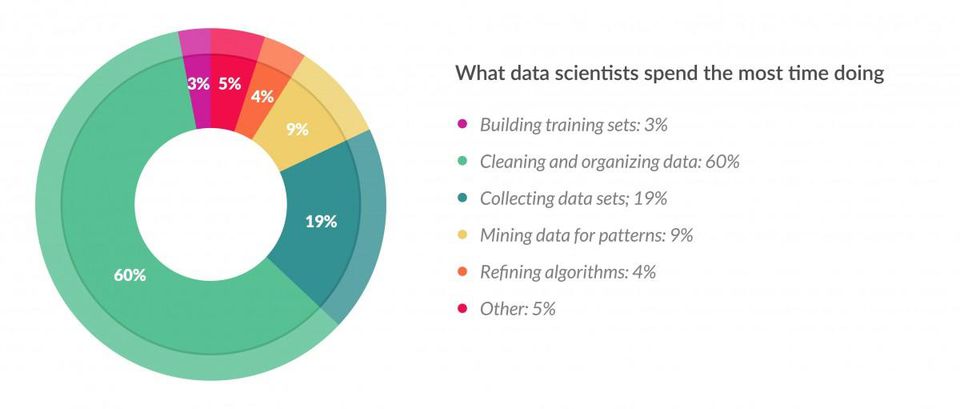
**3. Solution Approach (Main Idea)**

* **아이디어란주어진문제를효과적인체계적으로풀어나가기위한일련의처리과정을기술하는것입니다. What is your main idea to the problem?**

야구에서의 데이터 분석은 팀의 강점, 약점을 분석하고 강점은 더 활용하고 약점은 보완해 승리를 하기 위해 시행한다. 롤에서도 마찬가지로 결국 승리를 위해(problem) 데이터 분석을 시행한다.

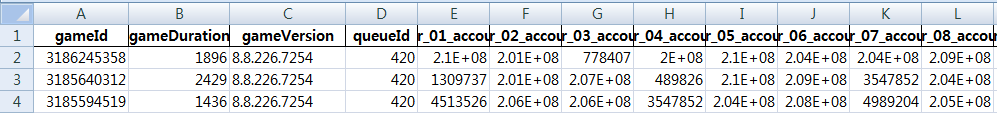
* 게임을 하다 보면 내가 A라는 챔피언을 픽 했을 경우, 우리 팀원이 B 나 C를 픽했으면 할 때가 있다. 이는 B나 C가 A와 조화가 잘 이루어질 거라 생각하기 때문이다. 이와 같이 이번 분석을 통해 어떤 챔피언 조합이 유용한지 알아내보려고 한다. (1). 2018년 4월에 관련 분석글이 올라왔다.)
* 일반적인 상황에서 어떤 챔피언을 고르는게 좋을지 분석을 통해 알아보려고 한다. 이는 그림2의 op.gg 챔피언 등급과 유사한데 나만의 알고리즘을 통해 이 등급을 산출할 것이다. 챔피언별로 (2).Elo rating system을 적용해 각각의 rating을 구해내고 등급화할 것이다.
* 챔피언별 승률의 평균, 픽률의 평균, 그리고 분산까지 기술통계를 작성해볼 것이다. Boxplot 또한 활용할 것이다.

이와 같은 데이터분석은 데이터 수집, 데이터 전처리및 가공, 모델링및 데이터 분석, 데이터 시각화및 보고서 작성의 과정으로 이루어진다. 데이터수집 그리고 이를 전 처리하고 가공하는 단계는 분석 소요 시간의 80%를 차지한다(그림 4 첨부). 이는 좋은 분석을 위해 질 좋은 데이터를 이용해야 되므로 성능을 위한 매우 중요한 단계이다. (1). 링크의 api를 이용해 필요한 데이터를 추출할 수 있었다. 롤의 게임 데이터는 하루에 \* GB 이상 생기는 만큼 방대하다. 따라서 필요한 데이터만 api를 이용해 추출하고 저장해야만 한다. 추출한 데이터는 json file type으로 no-sql형식이었는데 이를 sql즉 table 형식으로 변형해야 됐다. 필요한 모든 데이터를 table 형식으로 저장하고 나면 기술통계는 간단히 산출해낼 수 있다. 이제 모델링을 해야 하는데 ELO rating system 알고리즘을 이용할 데이터에 맞게 개발해야 한다. 챔피언 조합 또한 챔피언끼리 묶은 후 일정 이상 수의 표본을 기준으로 분석을 시행해야 한다.



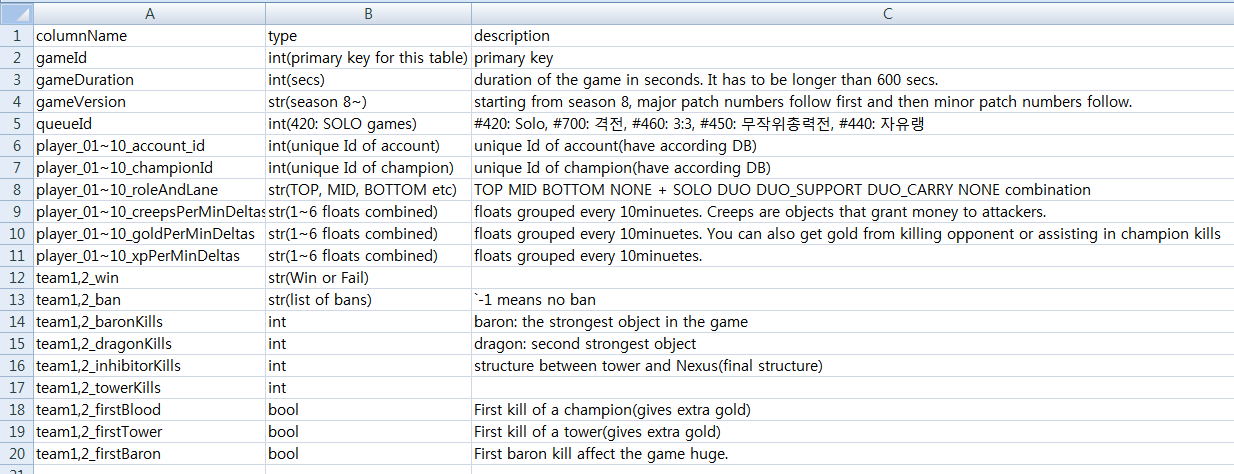
**<그림4 data analysis 시간소요비율>**

**데이터수집및전처리과정이80%를차지한다.**



**<그림5 .한국지역SOLO GAMES Example>**

**챌린저, 마스터리그인원의랭크게임데이터를모았다. gameId를primary key로총12500 \* 82 의데이터가있다.**



**<그림6 .data description>**

**82개의열을요약한표이다. 6~11은각각10개씩, 12~20은각각2개씩있다.**

**4. Experiment / 성능 평가**

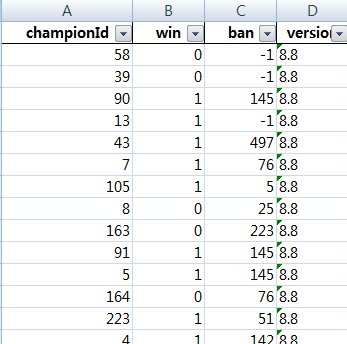
* **가설만들기(Hypotheses) / 가설검증/ 성능평가를위한실험계획**

챔피언간 등급을 알고리즘을 통해 산출해내고 챔피언간 조합에 따라 승률이 얼마나 나올지 예측해 본다. 이 예측이 얼마나 잘 맞을지 검증해보기 위해서 추후에 생기는 새로운 데이터를 이용할 것이다.

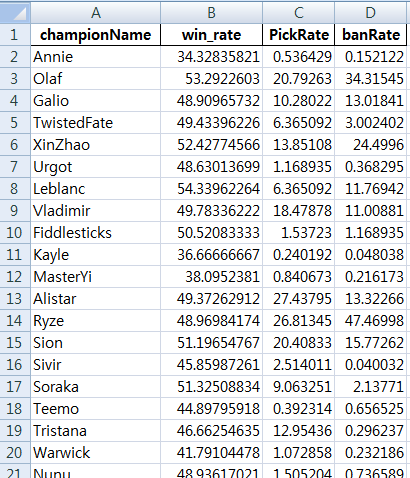
게임에서의 성능 평가는 간단하다. 추후에 있을 게임에서 승률이 얼마나 나왔는지를 가지고 알고리즘 및 분석을 통해 예상한 승률과 비교하는 것이다. 만약 새로운 게임 데이터를 구하기 힘들다면 최신 전적 10%를 기준으로 train 과 test dataset을 나누면 된다. 나머지 90%를 train dataset으로 이용하고 여기서 얻어진(train 한) 분석으로 test dataset에 적용해본다. 승률의 차이를 기준으로 loss function을 만들면 loss function이 제일 작은 것을 좋은 성능이 나왔다고 평가할 수 있다.

통계적 분석:

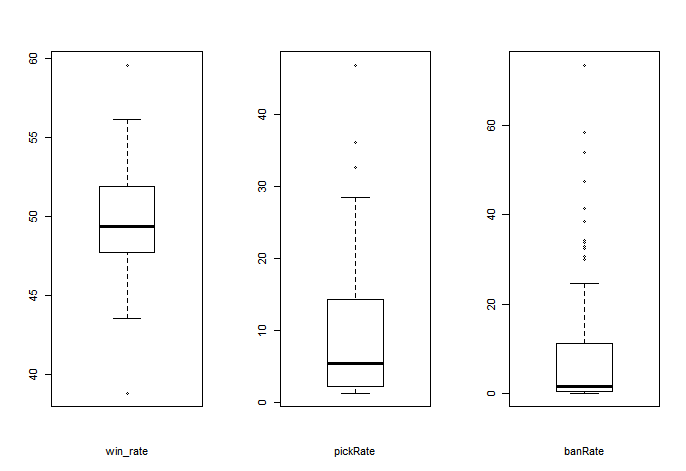
1. KR\_SOLO\_gameDataTable2.xlsx 에서부터 championId, win, ban, version 속성을 뽑아와 테이블로 만들었다. 이는 champion\_win\_ban.xlsx로 저장했다. row는 124900개가 있고 이는 총 12490게임에 대한 데이터이다.



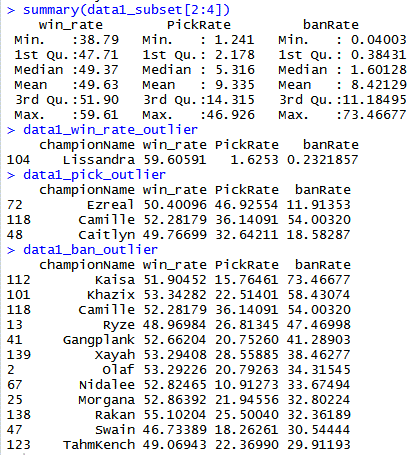
**2.** champion\_win\_ban에서 championId로 묶어 win\_rate, pick\_rate, ban\_rate를 계산했다.이는 champion\_rate.xlsx로 저장했다.

****

**3.** champion\_rate 으로부터 win\_rate, PickRate, banRate에 대해 boxplot을 그렸다.

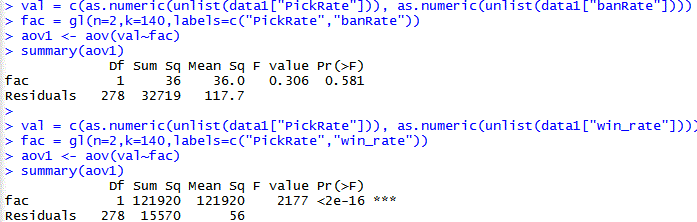
****

**4. boxplot에 대한 outlier를 구해봤다.**

****

데이터 중 표본 수가 작은 경우(Pick Rate 기준 1사분위 아래의 것)을 제외 했다. "리산드라"는 픽률1.6%에 승률 59.6%로 작은 픽률이지만 높은 승률을 기록했다. 정말 특수한 상황이거나 "리산드라"를 특별히 잘 다루는 이가 쓰기에 좋아 보인다. "이즈리얼", "카밀", "케이틀린"의 경우 모두 픽률이 30%를 넘겨 픽\_outlier로 뽑혔다. 이들 승률도 50%근방, 특히 "카밀"의 경우 52.3%나 되어 "카밀"은 어떤 상황에도 좋은 픽으로 보인다. "카밀"은 상대하기 까다로울 때 쓰는 밴에서도 54%를 기록했는데 픽률과 도합 84%로 140가지 중 10가지 챔피언만 나오는 게임에서 비중이 엄청 크다. Ban\_outlier의 경우 "카이사", "카직스", "카밀", 등 12가지가 있다. 밴의 경우 상당히 편차가 커서 outlier 또한 많은 것 같다. (즉 밴 당하는 챔피언만 많이 당하고 나머지는 median 1.6%에 해당하는 만큼 당한다)

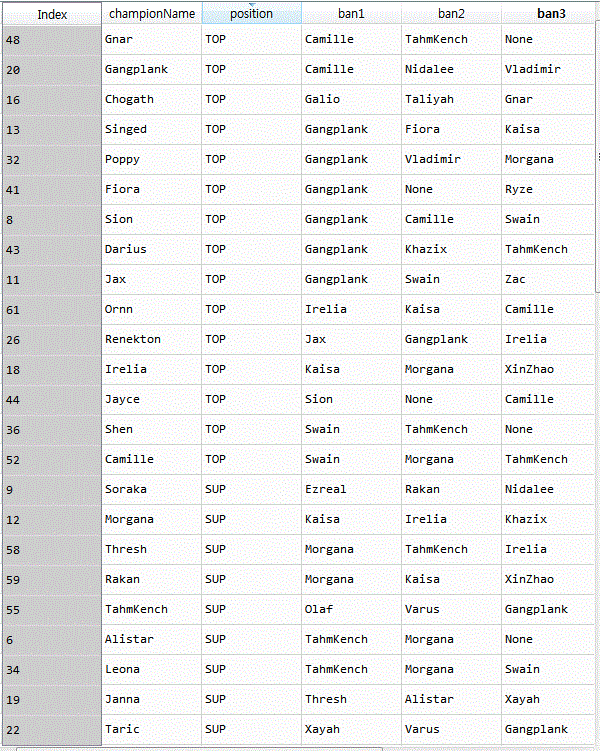
5. PickRate와 win\_rate 사이에 관련이 있을까 조사해봤다. ANOVA를 통해 검증해봤다.



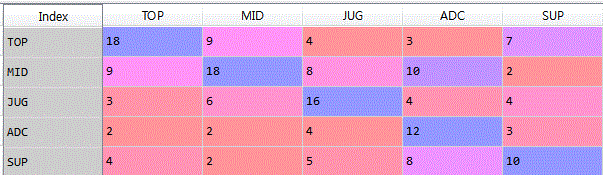
3번의 박스플롯에서 볼 수 있듯이 픽률과 밴률의 분포는 비슷하다. 두 집단 사이 관계도 p-value 값이 0.581로 두 집단이 같다는 null-hypothesis를 reject할 수 없다. 즉 이 두 집단은 같은 분포로부터 나왔을 수 도 있다. 상관관계는 0.69로 매우 높다.

반면에 픽률과 승률의 경우 P-value가 매우 작다. 즉 두 집단은 다른 분포로부터 나왔다고 볼 수 있다. 상관관계는 0.15다.

6. 다음으로 championName 별로 어떤 밴을 많이 했는지 알아봤다. value\_counts을 할때 normalize를 통해 각 champion별, 전체데이터와 비교가 쉽게 만들었다. 이후 전체 데이터에 비해 각 champion만의 밴이 특별히 높은 세가지를 뽑아냈다.



7. 이후 포지션(TOP, MID, SUP 등 역할을 뜻한다)별로 어떤 포지션을 밴하는지 구해봤다.



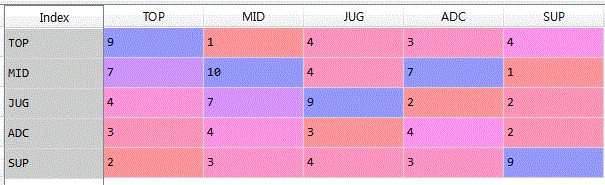
신기하게도 탑은 탑,미드를

미드는 미드,탑을

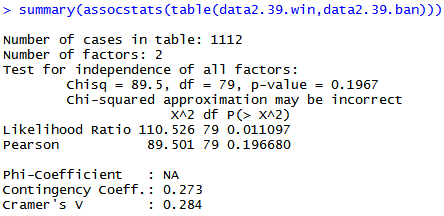
정글은 정글을

원딜은 원딜을

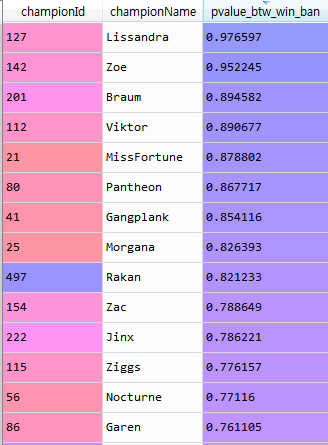
서폿은 서폿을 주로 밴했다. 즉 같은 포지션을 대상으로 밴을 많이 했다.

이는 8.8버전의 데이터에 한해서만 분석해본 결과이다. 버전 변수를 통제해도 마찬가지 경향이 보인다.

8. 각 챔피언별로 특정 챔피언을 밴하는게 승리에 도움이 되는 요소일까? 이를 위해 각 챔피언별로 특정 챔피언 밴여부와 승률간 관계를 분석했다. R라이브러리의 vcd. assocstats 을 이용했다. 이는 numeric 변수와 categorical 변수간 상관관계를 분석해준다.



39번 챔피언에 대한 분석이다. Cramer's V값이 크면 이들간 관계가 있다고 볼 수 있다. 또한 chi-square test에서 p-value값이 0.05보다 크면 null hypothesis(independence between 밴, 승리)를 reject할 수 있다. 즉 둘 간의 상관관계가 있다고 볼 수 있다.



p-value가 높은 순으로 정렬해봤다. 이 수치가 클수록 특정 밴을 했을 때 승률이 확연히 올라갈거라 해석했다. 먼저 "Zoe"의 경우 "None", " XinZhao ", "Olaf"가 Top3 밴인데, 전체 승률 0.512에 비해 각각을 밴했을 경우, 승률은 0.513, 0.425, 0.508다. XinZhao를 밴했을 경우 승률이 확연히 내려가는데 XinZhao와 Zoe간 시너지가 있는 것으로 예측해볼 수 있다.

"Darius"의 경우 Top3 밴은 " Gangplank", " Khazix", " TahmKench" 인데 전체 승률 0.504에 밴했을 경우 승률은 0.5, 0.581, 0.583이다. 카직스와 탐켄치를 밴했을 경우 승률이 확연히 올라간 것을 확인해 볼 수 있다.

9. 어떤 챔피언이 좋은 챔피언일까 고민해봤다. 단순히 승률만 높다고 승률 순으로 정렬해서 챔피언을 나열하면, 지극히 낮은 픽률의 챔피언들이 나오게 된다. 이와 같은 낮은 픽률의 챔피언의 경우, 특정 사람들만 픽을 하고 숙련도가 높아 승률이 높게 나왔다고 볼 수 있다. 승률이 50% 미만인 경우, 무조건적으로 그 챔피언이 나쁘다고 볼 수는 없다. 픽률이 높고 승률이 50%에 근접하면 regression to the mean 과 같은 현상을 고려해야 한다. 전체 승률이 50%에 근접하는 것과 같다. 이와 같은 것을 고려해서 승률을 보정해봤다. 과정은 이렇다. (다소 게임에 대한 이야기가 나온다)

1. 픽률 3.7%이상만 추출(약 100번이상의 게임플레이 수만 count)

2. 밴픽률 계산 ( 픽률 + 밴률)

3. 승률차 계산 ( 승률 - 50%)

4. 밴 가중치 계산 (승률차 \* 밴율)

4번의 경우, 밴이 안됐을 때 소환사가 픽을 한 경우 더해질 승률을 계산한 것.

일반적으로 특정챔의 밴해야될 픽 구성에 밴을 안했을 경우 특정챔의 승률 증가

(코그모픽에 상대방의 이렐픽). (코그모는 돌진에 취약)

50%이하의 승률을 보여도 4번의 절대값을 더해줘서 보정승률을 계산

5.보정승률 계산 (승률 + 밴가중치)

6.픽 보정 승률

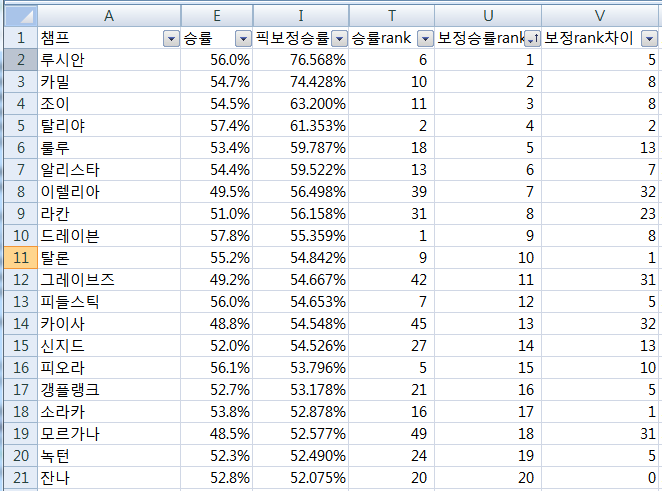
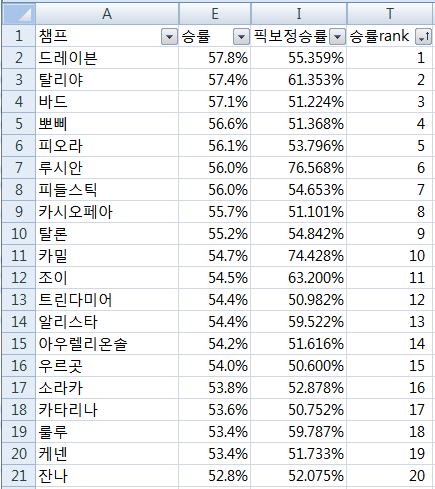
이번 문제 해결 알고리즘의 핵심

평균 픽률은 7.3%. 5\*2 / 137.

팀원수 \* 팀수 / 총 챔프수 (fow 기준 137개밖에 안나왔음)

평균 승률은 약 49.8%로 50%처리함.(fow 기준 생략된 data가 있어보임)

(보정 승률 \* 픽률 + (평균픽률\*2 - 픽률) \* 평균승률(50%)) / (평균픽률\*2)

** **

8.10버전 기준 보정 승률 rank로 나열해보면 루시안, 카밀, 조이, 탈리야, 룰루가 TOP5로 뽑히지만

단순 승률 rank로 보면 드레이븐, 탈리야, 바드, 뽀삐, 피오라가 TOP5로 뽑힌다.

전문가(챌린저) 기준 각각의 TOP5에서 좋은 챔피언을 고르라 하면 위의 경우 모두 뽑겠지만, 아래의 경우 탈리야만 뽑힐 것이다.

**5. Conclusion**

* **Summary / Contributions / Future work**

분석에 필요한 데이터를 api을 이용해 추출했다. nosql의 데이터중에서 이용할 데이터만을 골라 table 형식으로 변환해야 됐다. 이 데이터를 이용해 챔피언간 등급을 자체 지표 개발을 통해 산출해내고, 승률을 올려주는 챔피언 조합을 찾아낸다. 각 종 기술통계적인 내용을 시각화 해본다.

아직 연구가 많이 안된 분야인 좋은 챔피언 조합을 데이터를 통해 분석하려고 한다. 또한 챔피언간 등급의 경우 ELO rating system 으로 계산해내려고 한다. 이 방식을 이용하면 승률이 50%이하인 경우 높은 평가를 받을 수 없다.

게임 데이터 분석의 경우 필요한 데이터만 빠짐없이 있다면 정말 잘하는 코치들을 뛰어넘는 분석을 해서 전략을 보완할 수 있다. 또한 게임사의 경우 이러한 분석을 통해 균형 있는 게임을 만들어 낼 수 있다. 균형있게 게임을 바꾸어나가도 갑작스런 전략의 개발 등으로 균형이 무너지기 쉽다. 이를 데이터 분석을 통해 누구보다 빠르게 빈틈을 파고들어야 한다. 다음 데이터 분석의 경우 알고리즘의 속도까지 고려해 개발을 하고 싶다.

이번 분석에서 요즘 트렌드인 기계학습 알고리즘을 적용해 승리에 핵심적인 요인들을 분석해 보고 싶었다. 그러나 내가 가진 데이터는 너무 sparse했다. 기계학습을 이용하려면 더 많은 데이터를 수집해야 했다. 그 이용 방법으로는 어떤 챔피언이 특정 시간대에 어떤 아이템을 갔을때 승리에 도움을 주는지 예측해보는데 쓸 수 있었다. 그러나 챔피언으로 filtering하고 아이템, 시간대로 filtering하고 나면 고작 12600개 게임 중에서는 10개 이하로 나오게 된다. 이를 보완하려면 표본 대상을 "마스터, 챌린저" 구간이 아닌 더 넓은 구간으로 설정해 구해야 한다. 데이터를 수집했다고 하더라도 이를 기계학습에 맞게 가공하는 작업이 만만치 않다. 이번 프로젝트에서는 데이터 수집에 힘을 썼다면 다음 프로젝트 때는 원하는 기계학습에 맞게 가공하는 작업에 더 시간이 들이고 싶다.

**6. 별첨1(Source, data files description)**

**#dataFiles**

**#accountIdDB.xlsx**

**region, summoner\_id(primary key), accountId(primary key)**

**#championID.xlsx**

**id(primary key), key, name, title**

**#gameIdDB.xlsx**

**region, gameType, gameId(primary key)**

**#KR\_RANKED\_SOLO\_5x5\_challengerList.xlsx**

**rank\_order, summoner\_name,summoner\_id(primary key), accountId(primary key), tier, LP ,wins, losses**

**#KR\_RANKED\_SOLO\_5x5\_data.txt**

**원본 json형식의 dataset**

**#KR\_SOLO\_gameDataTable2.xlsx 이 데이터를 이용해 분석을 시행할 것이다**

**main dataset. decription below(그림 6)**

**KR\_SOLO\_gameDataTableDescription**

**#champion\_rate.xlsx**

**KR\_SOLO\_gameDataTable2.xlsx 에서 승률, 픽률, 밴률 비율을 뽑아낸 데이터**

**#champion\_win\_ban.xlsx**

**KR\_SOLO\_gameDataTable2.xlsx 에서 챔피언별로 승리, 밴, 패치버젼을 뽑아낸 데이터**

**#code**

**#getChampionIDList.py**

**챔피언 ID list를 구하기 위한 api 호출**

**산출물: championID.xlsx**

**#getBestSummonersStats.py**

**region, gametype, tier별로 챌린저 마스터 리그 인원들의 리스트 뽑기**

**산출물: KR\_RANKED\_SOLO\_5x5\_challengerList.xlsx**

**#getAccountIDBySummonerID.py**

**summonerID로 AccountID 받기**

**산출물: accountIdDB.xlsx**

**#addAccountIDWithBestSummoner.py**

**account ID로 챌린저 마스터 리그 인원들의 리스트 뽑기**

**산출물: KR\_RANKED\_SOLO\_5x5\_masterList.xlsx**

**#getAllGameId.py**

**accountId를 이용해서 gameID를 얻기. 해당 플레이어들이 최근 플레이한 100~200경기의 gameID를 뽑을수 있다.**

**산출물: gameIdDB.xlsx**

**#splitGameID.py**

**GameID를 gameType, region 별로 나누어 저장하기**

**산출물: dataKRSolo.xlsx**

**#getGameDataFromGameID.py**

**gameID 리스트로부터 각 gameID의 정보 뽑아내기**

**산출물: KR\_RANKED\_SOLO\_5x5\_data.txt**

**#getSummonerName.py**

**accountId를 이용해서 SummonerName 뽑아내기**

**#gameDataDictToTable.py**

**json 형태의 파일을 input으로 하여 이를 변환하여 일정 형식의 table을 output으로 산출**

**산출물: KR\_SOLO\_gameDataTable2.xlsx**

**#descriptiveAnalysisV1.py**

**기술통계 분석.**

**산출물: champion\_win\_ban.xlsx, champion\_rate.xlsx**

**#championRelationV1.py**

**챔피언, 변수간 관계분석**

**#descriptive.R**

**변수간 상관관계 분석, ANOVA 분석, boxplot 그리기**

**#API** “https://developer.riotgames.com/api-methods/”

**#champion-mastery**

챔피언몇번play 했는지 볼 수 있음

**#champion**

챔피언에 대한 간략한정보

**#leauge**

"/lol/league/v3/challengerleagues/by-queue/{queue}" : 챌린저

"/lol/league/v3/masterleagues/by-queue/{queue}" : 마스터

"/lol/league/v3/leagues/{leagueId}" :

"/lol/league/v3/positions/by-summoner/{summonerId}" :summonerId로 leagueId 체크 가능

챌린저,마스터 리그 사람들 확인가능

summeonrID로 티어 확인가능

챌린저 200명

마스터 201위~980위 정도(약770명)

**#lol-static-data**

정보 많음

/lol/static-data/v3/champions: 챔피언키들 #확인

/lol/static-data/v3/items: 아이템키들, 아이템목록들

/lol/static-data/v3/languages : 언어별 키들,#language-strings도 비슷. 쓸데없음

/lol/static-data/v3/maps :맵 그림파일

/lol/static-data/v3/masteries: 시즌7이전 마스터리 설명

/lol/static-data/v3/realms: 지역별 패치버젼확인

/lol/static-data/v3/reforged-rune-paths: 시즌8이후 룬 경로

/lol/static-data/v3/reforged-runes: 시즌8이후 룬, 설명

/lol/static-data/v3/runes: 시즌7이전 룬 설명 #필요없음

/lol/static-data/v3/summoner-spells: 스펠 id, 설명

/lol/static-data/v3/versions: 버젼확인 #realms와 겹침

**#lol-status**

서버 상태 확인

**#match**

##이게 중요

1. match 정보 matchID를 통해 얻을 수 있음.

2. 150개정도 최근 match 기록 확인가능

3. 최근20개 match 기록확인가능

4. 1분간격으로 레벨, 위치 기록. 템이나cs 확인가능. Event발생 확인가능

**#matches**

#gameId(int) ##Primary key

#gameDuration(int)

#gameVersion(str)

#participantIdentities(list) ##size:10

#dict

#participantId(int) : 1~10 ##for team differentiation

#player(dict)

#accountId(int)

#participants(list) ##size 10

#dict

#championId(int)

#participantId(int)

#spell1Id(int)

#spell2Id(int)

#stats(dict)

#masteries included

#teamId ##100 for 1 to 5, 200 for 6 to 10

#timeline(dict) ##grouped by every 10 minutes

#role ##SOLO, DUO\_CARRY, NONE(JUNGLE), DUO\_SUPPORT

#lane ##TOP, BOTTOM, JUNGLE, MID

#creepsPerMinDeltas(dict) ##"0-10", "10-20", ...

#goldPerMinDeltas(dict)

#xpPerMinDeltas(dict)

#queueId(int) ## defines gameType. Better use this gameType

#420: Solo

#700: 격전

#460: 3:3

#450: 무작위총력전

#440: 자유랭

#teams(list)

#dict ##100, 200

#win(str) ##Win, Fail

#bans(list) ##-1: no ban

#baronKills(int)

#dragonKills(int)

#inhibiorKills(int)

#towerKills(int)

#riftHeraldKills(int)

#firstBaron(bool)

#firstBlood(bool)

#firstDragon(bool)

#firstInhibitor(bool)

#firstRiftHerald(bool) ##바위게

#firstTower(bool)

#timelines ##frameInterval(1 minutes)(60000ms)

#events(list)

#dict

#type ##main differentiable key

#timestamp

#participantId

#killerId

#victimId

#assistingParticipantIds

#position

#itemId

#...etc

#participantFrames(dict) ##every minute ex). 0:00, 1:00, ...,

#participantId ##primary key

#totalGold

#currentGold

#level

#xp

#minionsKilled

#jungleMinionsKilled

#position(x,y)

#timestamp(int) ##exact time(ms)

**#spectator**

summonerID로 입력받는데

잘 안됨

**#summoner**

summonerID

accountID

summonerName 따로 있음

우리가 일반적으로 아는 아이디는 summonerName