빅데이터 분석 및 시각화 개론 기말 발표

♚♚히어로즈 오브 더 스☆톰♚♚

가입시$$전원 카드팩☜☜뒷면100%증정※
♜월드오브 워크래프트♜펫 무료증정￥
특정조건 §§디아블로3§§★공허의유산★초상화획득기회@@@

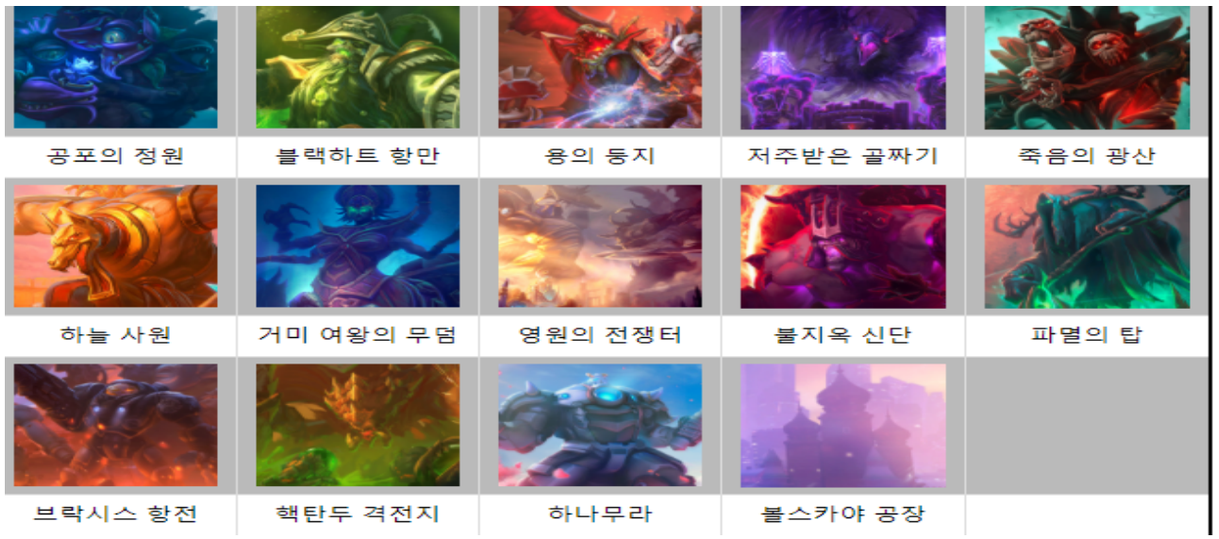
20조 권오현, 김승년, 허재형

Index

HotS?

1. 5:5 팀전 게임 - 이기거나 질 수 있음

2. 20가지의 전장이 존재



3. 65종류의 영웅을 선택할 수 있음

Data

1. https://www.hotslogs.com/Info/API

2. 11월 8일 기준 30일 전 &전 세계 서버

3. 3가지 데이터

HeroIDandMAPID.csv

Out [3] :

	ID	Name	Group	SubGroup
0	0	Unknown	NaN	NaN
1	1	Abathur	Specialist	Utility
2	2	Anub'arak	Warrior	Tank
3	3	Arthas	Warrior	Bruiser
4	4	Azmodan	Specialist	Siege

ReplayCharacters.csv - 950만개의 row

Out [4] :

	ReplayID	Is Auto Select	HeroID	Hero Level	Is Winner	MMR Before	In Game Level	Takedowns	Killing Blows	Assists	Deaths
0	123508648	0	42	14	1	1978.0	24	49	3	46	1
1	123508648	0	52	12	1	2037.0	24	51	15	36	8
2	123508648	0	68	3	0	2021.0	24	18	7	11	17
3	123508648	0	59	10	0	1886.0	24	19	3	16	13
4	123508648	0	40	20	1	2046.0	24	51	25	26	7

Replays.csv

Out [5] :

	ReplayID	GameMode(3=Quick Match 4=Hero League 5=Team League 6=Unranked Draft)	MapID	Replay Length	Timestamp (UTC)
0	123508648	3	1001	00:31:59	9/22/2017 3:01:18 AM
1	123826826	3	1003	00:22:32	9/22/2017 3:01:22 AM
2	123789056	3	1007	00:21:24	9/22/2017 3:01:23 AM
3	126610491	3	1001	00:15:52	9/22/2017 3:01:24 AM
4	123497937	3	1008	00:22:55	9/22/2017 3:01:25 AM

Data

	A	B	C	D
1	153846.2		승률	0.5
2	76923.08	기댓값	패배율	0.5
3	38461.54	분산		
4	196.1161	표준편차		
5	1177	6시그마		
6	75746	0.492351		
7	78100	0.507649		

표본의 6시그마 평균 분포는 49.2%~50.8%

만약 특정한 분석 결과가 이 범위를 벗어나는 경우,

그때는 그 승률이 평균을 넘어서는 유의미한 결과

Experience

1. Hero Level = 영웅의 숙련도(경험)
2. 영웅의 숙련도가 승률에 큰 영향(가설)

‘is winner’와 ‘Hero Level’ data

```
In [7]: # 일단 Replay_character 파일에서 is winner 랑 Hero Level이랑 묶기
        Rep_Char01 = Rep_Char1[['Is Winner','Hero Level']]
        Rep_Char01.head()
```

```
In [5]: h=0;
        for i in range(0,10000):
            b = 10*i + 10;
            a = Rep_Char01[10*i:b]
            a1 = a[a['Is Winner'] == 0]
            a2 = a1['Hero Level'].sum()
            a3 = a[a['Is Winner'] == 1]
            a4 = a3['Hero Level'].sum()
            if a2<a4:
                h = h+1
            else:
                h = h
        h
Out[5]: 5282
```

- 1만개의 Sample 기준 -> 52.82%
- 95만개의 Sample(전체) 기준 -> 53.7%(49/95)

숙련도가 높은 쪽이 이긴 경우가 53%.
: 진 경우와 비교할때 5-6%p차이.
통계적 평균의 범위를 벗어남 -> 유의미.

Damage

1. 피해량(캐릭터/구조물)과 승률(가설)
->'Damage'와 'is winner' data

```
In [10]: Rep_Char02.head()
```

Out[10]:

	ReplayID	Is Winner	Hero Damage	Siege Damage	total Damage
0	123508648	1	16313	9331	25644
1	123508648	1	89553	28938	118491
2	123508648	0	74827	53071	127898
3	123508648	0	54413	38493	92906
4	123508648	1	78745	50724	129469

- 한 판당(ReplayID) winner와 loser의 피해량 비교

```
In [12]: k.columns = ['loser Hero Damage', 'winner Hero Damage','loser siege Damage', 'winner siege Damage', 'loser total D
         k.head()
```

Out[12]:

	loser Hero Damage	winner Hero Damage	loser siege Damage	winner siege Damage	loser total Damage	winner total Damage
0	329668.0	413495.0	434299.0	379951.0	495657.0	463778.0
1	234097.0	289378.0	289900.0	294205.0	353124.0	349486.0
2	169197.0	185139.0	286803.0	239916.0	328679.0	255858.0
3	133938.0	150984.0	198815.0	143811.0	226695.0	160857.0
4	164136.0	190141.0	217649.0	260033.0	248031.0	286038.0

```
In [15]: k1 = k.head(1000)
         k1.describe()
```

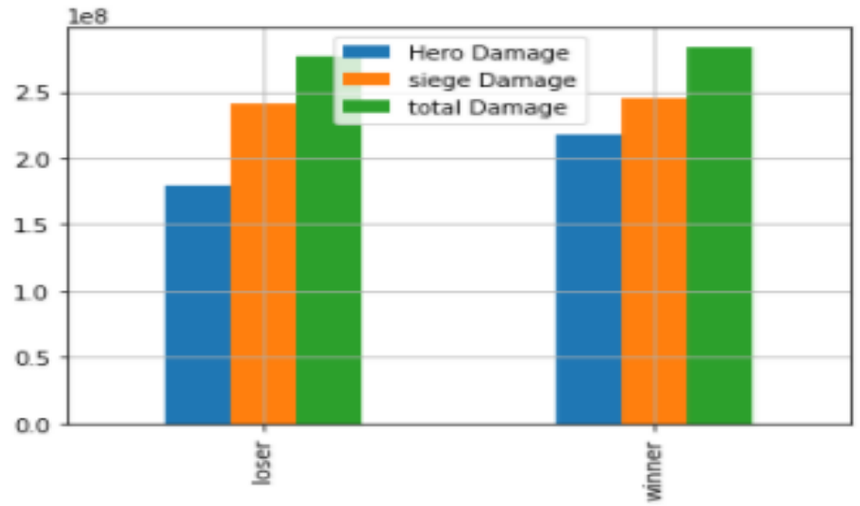
Out[15]:

	loser Hero Damage	winner Hero Damage	loser siege Damage	winner siege Damage	loser total Damage	winner total Damage
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	179177.048000	217697.836000	241209.252000	245637.640000	276472.771000	284158.428000
std	67335.021804	77805.853609	90884.840496	90686.647266	104858.136914	101046.922747
min	20277.000000	37578.000000	56389.000000	54205.000000	63644.000000	71506.000000
25%	128440.250000	160318.750000	173999.250000	177947.500000	200118.250000	208957.750000
50%	169695.500000	205944.500000	229175.000000	232666.500000	260906.500000	270730.500000
75%	219425.250000	263204.250000	290575.250000	297276.000000	333228.750000	338802.250000
max	473458.000000	542962.000000	641930.000000	643446.000000	728436.000000	728571.000000

Damage

피해량에 대한 분석결과

```
In [31]: k3.plot(kind='bar')
         plt.grid()
```



Winner와 Loser의 캐릭터 피해량 차이는 거의 20%
>> 캐릭터 피해량이 구조물 피해량보다 승패에 더 기여

또한, 피해량이 영웅 숙련도보다 더 승리에 많이 기여

이에 따라 피해량에 대한 세세한 분석이 필요.
또한, 피해량뿐만 아니라 다른 수치에 대해서도 분석해야 함.
맵별로 다른 경향성을 보일 수 있기에 맵별로 나누어 분석.

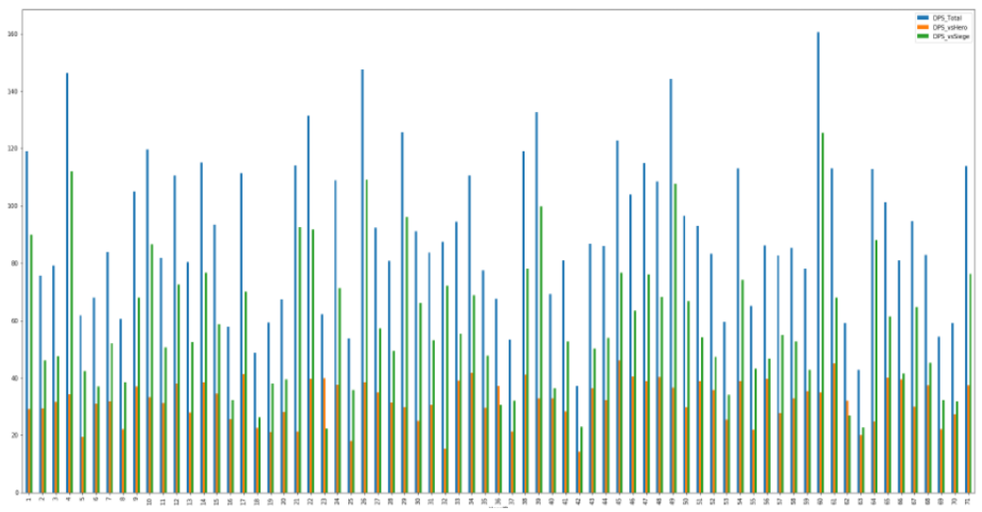
Efficiency

저주받은 골짜기(1003) : 가장 기본적인 맵

```
mapID = 1001      # 1001 ~ 1020
want = 1          # 0: DPS_Total, 1: DPS_vsHero, 2: DPS_vsSiege, 3: DPS_DMG_Taken, 4: DPS_Healing
tmp = ""

a=Rep
a1 = a[a['MapID'] == mapID]
b = pd.merge(Rep_Char, a1, how='inner', left_on = 'ReplayID', right_on='ReplayID')
b['Alive Time']=0
for i in range(0, 10000):
    b['Alive Time'][i]=(datetime.datetime.strptime(b['Replay Length'][i], "%H:%M:%S") - datetime.datetime.strptime(b['Time Spent Dead'][i], '
b_sampled = b[0:10000]
b_sampled['Total_Damage'] = b_sampled['Hero Damage']+b_sampled['Siege Damage']

b_sampled['DPS_Total'] = b_sampled['Total_Damage']/b_sampled['Alive Time']
b_sampled['DPS_vsHero'] = b_sampled['Hero Damage']/b_sampled['Alive Time']
b_sampled['DPS_vsSiege'] = b_sampled['Siege Damage']/b_sampled['Alive Time']
b_sampled['DPS_DMG_Taken'] = b_sampled['Damage Taken']/b_sampled['Alive Time']
b_sampled['DPS_Healing'] = (b_sampled['Healing']+b_sampled['Self Healing'])/b_sampled['Alive Time']
dps_by_hero = b_sampled[['HeroID', 'DPS_Total', 'DPS_vsHero', 'DPS_vsSiege', 'DPS_DMG_Taken', 'DPS_Healing']].groupby('HeroID')[['DPS_Total',

if want ==0 :
    tmp = 'DPS_Total'
elif want == 1:
    tmp = 'DPS_vsHero'
elif want == 2:
    tmp = 'DPS_vsSiege'
elif want == 3:
    tmp = 'DPS_DMG_Taken'
elif want == 4:
    tmp = 'DPS_Healing'
NO1 = dps_by_hero[tmp].sort_values(ascending = False).index[0]
Hero_Map[Hero_Map['ID']==NO1]['Name']
```

2) 딜/힐/탱을 잘 할수록 팀 승리에 더 높은 기여(가설)
->이는 딜량 분석으로 증명
->후에 기여도 함수를 만든다면 넣을 수 있음.

```
In [67]: dps_by_hero.plot.bar(figsize=(30,15))
Out[67]: <matplotlib.axes._subplots.AxesSubplot at 0x2404dccdf28>
```



Efficiency

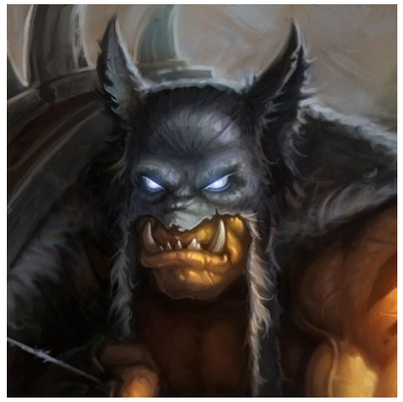
Total Damage



Hero Damage

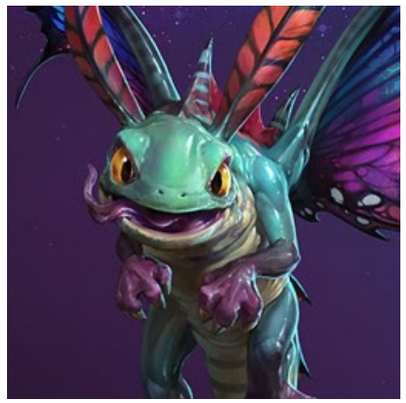


Damage Tank


Heal





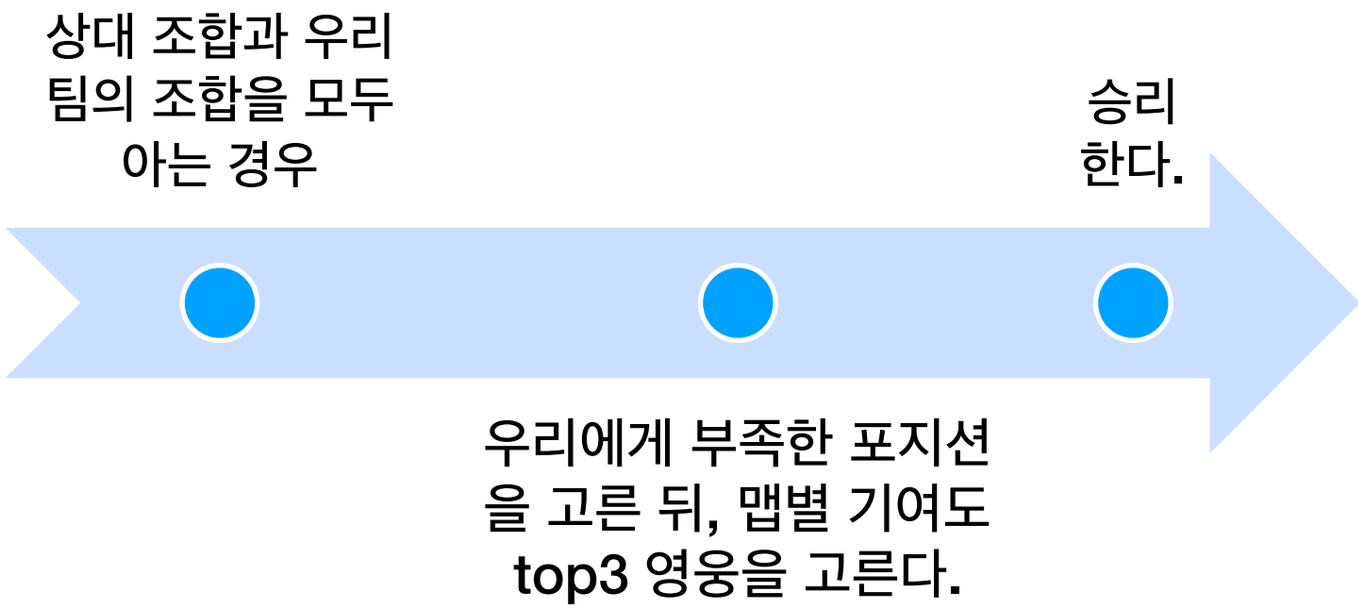
To Win

1. 최적의 조합?

1) 조합의 중요성

2) 챔피언을 고르면 승률을 계산하는 기여도 방정식 필요

2. Plan A – 최적의 영웅 선택



To Win

3. Plan B – 승률 방정식(추후)

1) 우리 팀 5명의 픽이 정해지지 않아 조합을 찾는 경우

2) Logistic Regression을 통한 승률 방정식 제작

3) 데이터가 부족하여 현실적으로는 불가(10개로 실행)

Y(승리여부 – 0 or 1 // binary data)
= a(특정계수)*(선택한 챔피언들의 딜량/탱킹/힐링 순위의 합) +b(특정계수)*(카운터 픽의수) +c(특정계수)*(조합 시너지)

-> a, b, c를 계산하기!

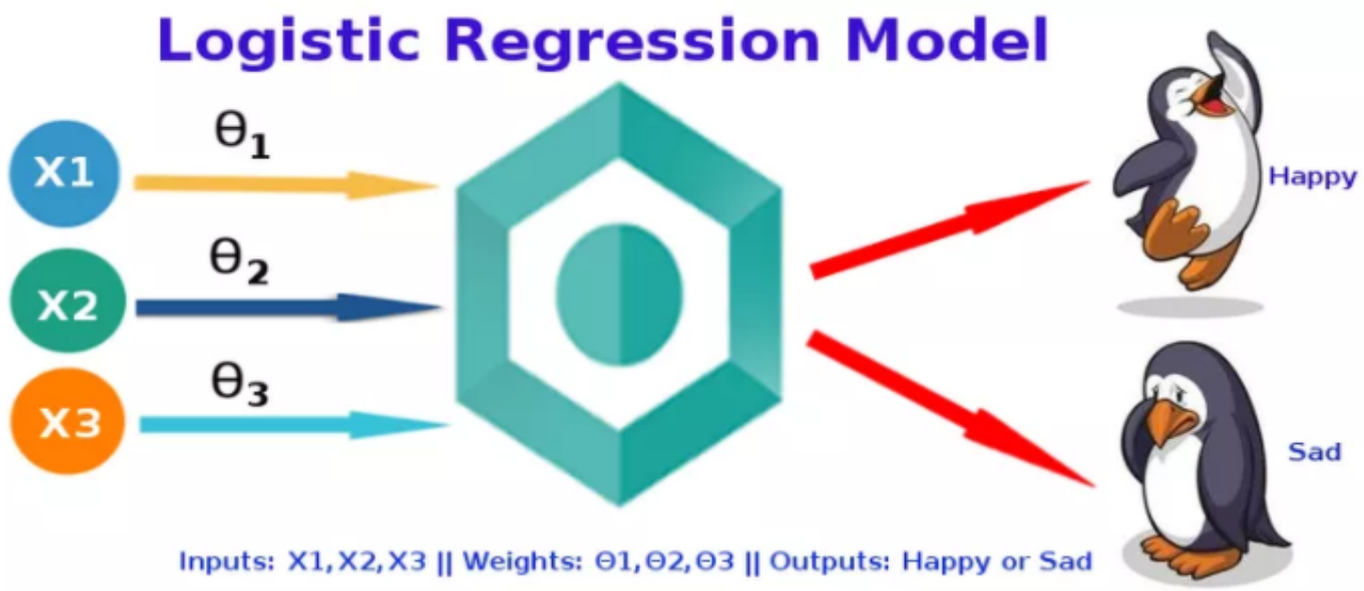
*새로운 변수

1) 카운터 픽-> 이전 데이터에서 a라는 챔프를 골랐을 때, 상대에게 b라는 챔프가 있을 때, 승률이 50을 넘지 않으면 카운트 1
2) 2)조합 시너지-> 자주나온 정도, 전체 데이터에서 실제 이 조합이 나온 횟수

Regression

3. Plan B – 승률 방정식(추후)

4) Logistic Regression



$$h_\theta(x) = g(\theta^T x)$$

Sigmoid function으로 가정

```
#SIGMOID Compute sigmoid functoon
#   J = SIGMOID(z) computes the sigmoid of z.

# You need to return the following variables correctly
g = np.zeros(z.shape)

# ============== YOUR CODE HERE ==============
# Instructions: Compute the sigmoid of each value of z (z can be a matrix,
#               vector or scalar).

g = (1/(1+np.exp(-z)))

# =============================================

return g
```

Regression

3. Plan B – 승률 방정식(추후)

5) Cost Function // SGD

```
for i in range(m):

    J = J + (-y[i]*math.log(sigmoid(np.dot(th.T,X[i])))-(1-y[i])*math.log(1-sigmoid(np.dot(th.T,X[i]))))/m

# ============================================
```

```
In [7]: def gradient(X, y, th, _lambda=0):
            # Initialize some useful values
            m = len(y)

            # You need to return the following variables correctly
            grad = np.zeros(th.shape)

            # ============= YOUR CODE HERE ==============

            for i in range(m):

                grad = grad + ((sigmoid(np.dot(th.T,X[i]))-y[i])*X[i])/m

            # ============================================

            return grad
```

87%

```
In [9]: cost(X, y, np.array(initial_theta))
Out[9]: 0.6931471805599606
```

Cost(오차)

```
In [12]: optimal_theta
Out[12]: array([-24.40141901,   0.2001471 ,   0.19532949])
```

A,B,C값

Regression

3. Plan B – 승률 방정식(추후)

6) 정확도

In [14]:
```
def predict(t, x):
    #PREDICT Predict whether the label is 0 or 1 using learned logistic
    #regression parameters theta
    #   p = PREDICT(theta, X) computes the predictions for X using a
    #   threshold at 0.5 (i.e., if sigmoid(theta'*x) >= 0.5, predict 1)

    m = X.shape[0] # Number of training examples

    # You need to return the following variables correctly
    p = np.zeros(m)

    # ========== YOUR CODE HERE ==========
    # Instructions: Complete the following code to make predictions using
    #               your learned logistic regression parameters.
    #               You should set p to a vector of 0's and 1's
    #
    for i in range(m):
        a = (sigmoid(np.dot(t.T,x[i])))

        if a >= 0.5:
            p[i] = 1
        else:
            p[i] = 0

    # ==========================================

    return p
```

0.5 미만은 0
0.5 이상은 1
실제 코드에 대입해서
몇 퍼센트가 되는지 확인

89%

Let's predict the admission probably of a student with scores 45 and 85:

Training set accuracy:

In [15]:
```
print 'logistic regression, training accuracy : ' +str(np.mean(predict(optimal_theta, X) == y))
```

```
logistic regression, training accuracy : 0.89
```

Synergy

3. Plan B – 시너지 효과의 계산

In [199]:
```
k4.iloc[0]['hero5']
```

Out[199]:
```
71
```

In [204]:
```
tmp = [2, 16]
k6 = pd.DataFrame(columns = ['ReplayID', 'hero1','hero2','hero3','hero4','hero5','isWin'])
k7 = pd.DataFrame(columns = ['ReplayID', 'hero1','hero2','hero3','hero4','hero5','isWin'])
count=0
for i in range(0, 20000):
    if tmp[0]==k4.iloc[i]['hero1'] or tmp[0]==k4.iloc[i]['hero2'] or tmp[0]==k4.iloc[i]['hero3'] or tmp[0]==k4.iloc[i]['hero4'] or tmp[0]==k
        k6.loc[count]=k4.iloc[i]
        count+=1

count2=0
for i in range(0, count):
    if tmp[1]==k6.iloc[i]['hero1'] or tmp[1]==k6.iloc[i]['hero2'] or tmp[1]==k6.iloc[i]['hero3'] or tmp[1]==k6.iloc[i]['hero4'] or tmp[1]==k
        k7.loc[count2]=k6.iloc[i]
        count2+=1
k7
```

Out[204]:

	ReplayID	hero1	hero2	hero3	hero4	hero5	isWin
0	123498069	2	16	51	62	65	0
1	123621834	2	16	39	63	71	1
2	123500623	2	16	22	36	48	0
3	123504207	2	16	34	52	62	0
4	123812400	2	16	38	51	70	1
5	127182194	2	16	25	52	71	1
6	124175597	2	16	39	65	69	0
7	124582156	2	5	15	16	34	0
8	128713507	2	16	30	46	67	1
9	123623646	2	11	14	16	51	1
10	123535128	2	12	16	53	56	1
24	125756325	1	2	16	43	47	1
25	124208927	1	2	5	16	39	1
26	124137896	2	16	21	36	38	0

In [205]:
```
z=k7['isWin'].sum()/k7['isWin'].count()
z
```

Out[205]:
```
0.5555555555555556
```

In [208]:
```
a=k5[k5['HeroID']==tmp[0]]['Is Winner'].sum()/k5[k5['HeroID']==tmp[0]]['Is Winner'].count()
b=k5[k5['HeroID']==tmp[1]]['Is Winner'].sum()/k5[k5['HeroID']==tmp[1]]['Is Winner'].count()
(a+b)/2>z #true면 시너지 낮음, false면 높음
```

Out[208]:
```
False
```

Verification

검증 데이터

(1) Cross Validation(k=5)를 사용

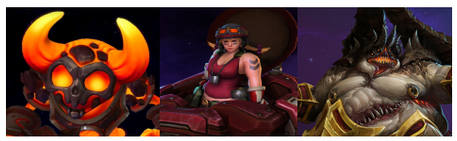
(2) 5만개의 데이터를 1만개씩 나누어 유사한지를 확인하였다. 원래는 오차율이 가장 적은 sample을 사용하지만 우리는 각기 데이터를 모두 받았다.

기본 데이터 top3

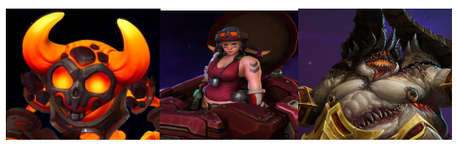
51% 55% 51%

Sample 1 기준

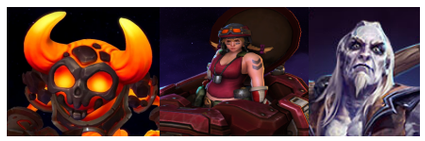
Sample 2 기준



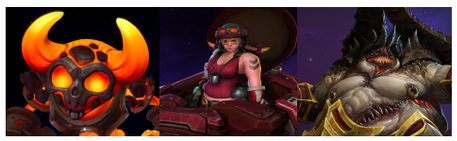
Sample 3 기준



Sample 4 기준



Sample 5 기준

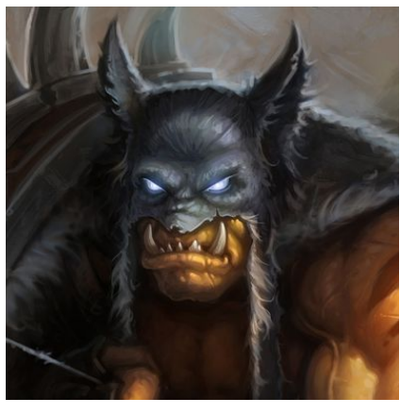


Verification

>50%

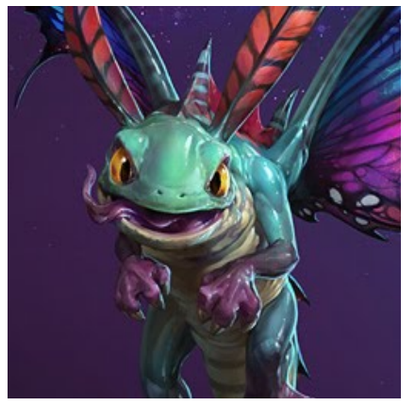


51%, 53%, 53%


>50%, 47%, 47%





Conclusion

분석 결과

(1) 많이 한 사람 = 숙련도가 더 높은 사람 = 경험이 더 많은 사람이 더 이기는 것을 확인하였다.

(2) 이긴 팀은 적 영웅에 유의미하게 딜을 더 많이 한 것을 볼 수 있다.

(3) Map에 따른 딜량/탱킹/힐량 top3를 찾았다.

(4) 적은 데이터지만 조합을 넣으면 승률을 계산하는 방정식을 만들었다. 두 영웅의 시너지 효과 역시 계산해 볼 수 있었다.

(5) 계산한 방정식을 통해, 모든 조합을 대입하면 가장 승률이 높은 조합을 알 수 있다.
-> 하지만 가짓수가 10억 가지가 넘어가 계산하기 힘들다.

♚♚히어로즈 오브 더 스☆톰♚♚

가입시$$전원 카드팩☜☜뒷면100%증정※

♜월드오브 워크래프트♜펫 무료증정￥

특정조건 §§디아블로3§§★공허의유산★초상화획득기회@@@

QnA