롤 티어 예측모델 만들기

이민수 (팀장! 겸 발표자! 겸 커피물조절장인)
오서영 (교수님!)
이수빈 (피피티!)

목차

데이터 설명

- OP.GG


솔랭
2일 전
승리
21분 1초
조이
3 / 2 / 5
4.00:1 평점
레벨13
146 (6.9) CS
킬관여 38%
매치 평균
Grandmaster
제어 와드 5
배호영
I dont kno...
IGzhao14...
arrowlol
Ye Zhuo ...
zyb
익산누누
T1 Fisher
GGGGali
SANDBO...
종합
팀 분석
빌드
etc
승리 (레드팀)
티어
OP Score
KDA
피해량
와드
CS
아이템
zyb Challenger 5.6 7th 1.17:1 3/6/4 (33%) 12,261 2 9 / 0 152 분당 7.2
익산누누 Challenger 5.9 6th 8.00:1 2/1/6 (38%) 3,968 1 2 / 2 141 분당 6.7
T1 Fisher Challenger 7.4 2nd 4.00:1 3/2/5 (38%) 14,372 5 11 / 2 146 분당 6.9
GGGGali Challenger 9 MVP 7.50:1 9/2/6 (71%) 15,579 2 8 / 7 204 분당 9.7
SANDBOX ... Grandmaster 7 4th Perfect 4/0/13 (81%) 6,968 7 24 / 4 26 분당 1.2
1 3 4
Total Kill 21 11
Total Gold 43567 37045
0 0 3
패배 (블루팀)
티어
OP Score
KDA
피해량
와드
CS
아이템
배호영 Challenger 7.3 ACE 2.67:1 2/3/6 (73%) 8,656 6 12 / 2 118 분당 5.6
I dont know ... Grandmaster 6.7 5th 2.33:1 4/3/3 (64%) 8,938 5 7 / 3 131 분당 6.2
IGzhao1498... Grandmaster 4.9 8th 1.00:1 3/5/2 (45%) 5,641 0 1 / 4 170 분당 8.1
arrowlol Challenger 4.3 10th 0.29:1 2/7/0 (18%) 6,867 1 6 / 4 159 분당 7.6
Ye Zhuo Yue Challenger 4.5 9th 1.00:1 0/3/3 (27%) 3,818 3 16 / 6 44 분당 2.1

```
csv = pd.read_csv('lol_dataset.csv', encoding = 'cp949')
```

```
csv.head()
```

	id	time	tier	KDA	DPS	ward	cs	cspm
0	1	31분 58초	Challenger	6/5/11	31367	0/10/5	296	9.3
1	2	31분 58초	Challenger	9/6/13	24054	7/11/8	141	4.4
2	3	31분 58초	Challenger	9/1/7	18880	2/10/15	377	11.8
3	4	24분 1초	Challenger	14/4/13	25399	8/16/1	108	4.5
4	5	24분 1초	Challenger	6/3/8	18421	4/7/4	194	8.1

2-1) time

```
time_orig = csv['time']
time_orig = np.array(time_orig)
time_orig[:5]
# time_orig.shape
```

```
time_minute = []
for i in range(len(time_orig)):
    mod = time_orig[i].split('분')
    time_minute.append(int(mod[0]))

print(time_minute)
```

'분'을 기준으로 split
함수를 이용
앞에 숫자만!

변수 생성

2-2) KDA

```
KDA_orig = csv['KDA']
```

```
KDA_K = []
KDA_D = []
KDA_A = []
for i in range(len(KDA_orig)):
    mod = KDA_orig[i].split('/')
#     print(i)
    KDA_K.append(int(mod[0]))
    KDA_D.append(int(mod[1]))
    KDA_A.append(int(mod[2]))


KDA_K = np.array(KDA_K)
KDA_D = np.array(KDA_D)
KDA_A = np.array(KDA_A)
```

Kill,Death,Assist
각각열로 나눠준다.

```
KDA_cal = []
for i in range(len(KDA_orig)):
    if KDA_D[i] == 0:
        KDA_D[i] = 1  ## inf 방지
    else :
        KDA_cal.append(int((KDA_K[i]+KDA_A[i])/KDA_D[i]))
```

(K+A)/D

```
KDA_cal = np.array(KDA_cal)
np.shape(KDA_cal)
```

```
(300,)
```

2-3) Ward

```
ward_orig = csv['ward']
```

```
ward1 = []
ward2 = []
ward3 = []
for i in range(len(ward_orig)):
    mod = ward_orig[i].split('/')
    ward1.append(int(mod[0]))
    ward2.append(int(mod[1]))
    ward3.append(int(mod[2]))
#     print(i)

ward1 = np.array(ward1)
ward2 = np.array(ward2)
ward3 = np.array(ward3)
```

```
ward_sum = ward1 + ward2 + ward3
```

2-4) DPS per minute

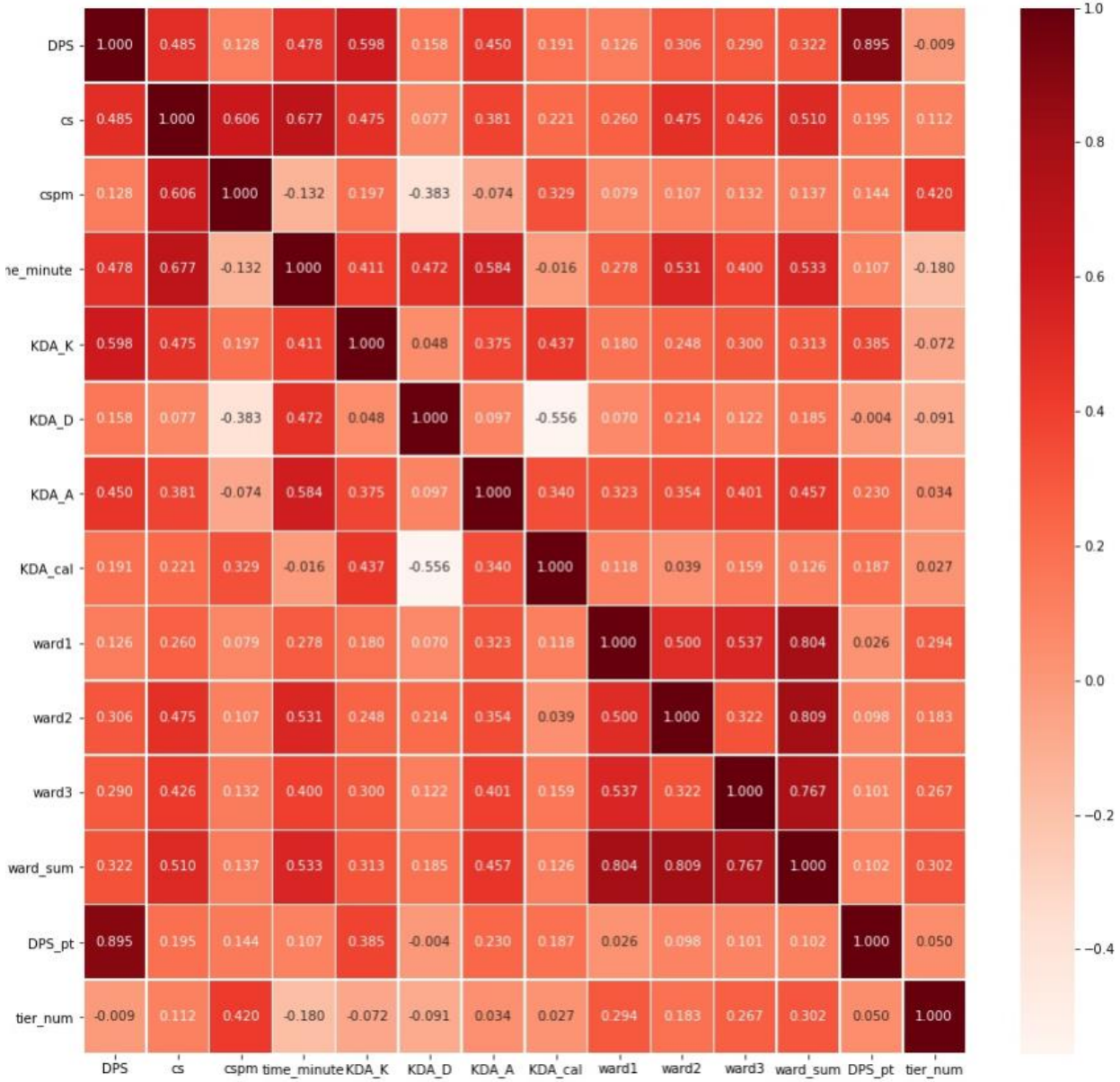
```
DPS_pt = DPS/time_minute
```

2-5) tier

```
tier_uni=csv['tier'].unique()
tier_uni
['Iron 4', 'Iron 3', 'Iron 2','Iron 1','Bronze 4','Bronze 3','Bronze 2','Bronze 1','Silver 4','Silver 3','Silver 2','Silver 1',
 'Gold 4','Gold 3','Gold 2','Gold 1','Platinum 4','Platinum 3','Platinum 2','Platinum 1','Diamond 4','Diamond 3','Diamond 2',
 'Diamond 1','Master','Grandmaster','Challenger']
```

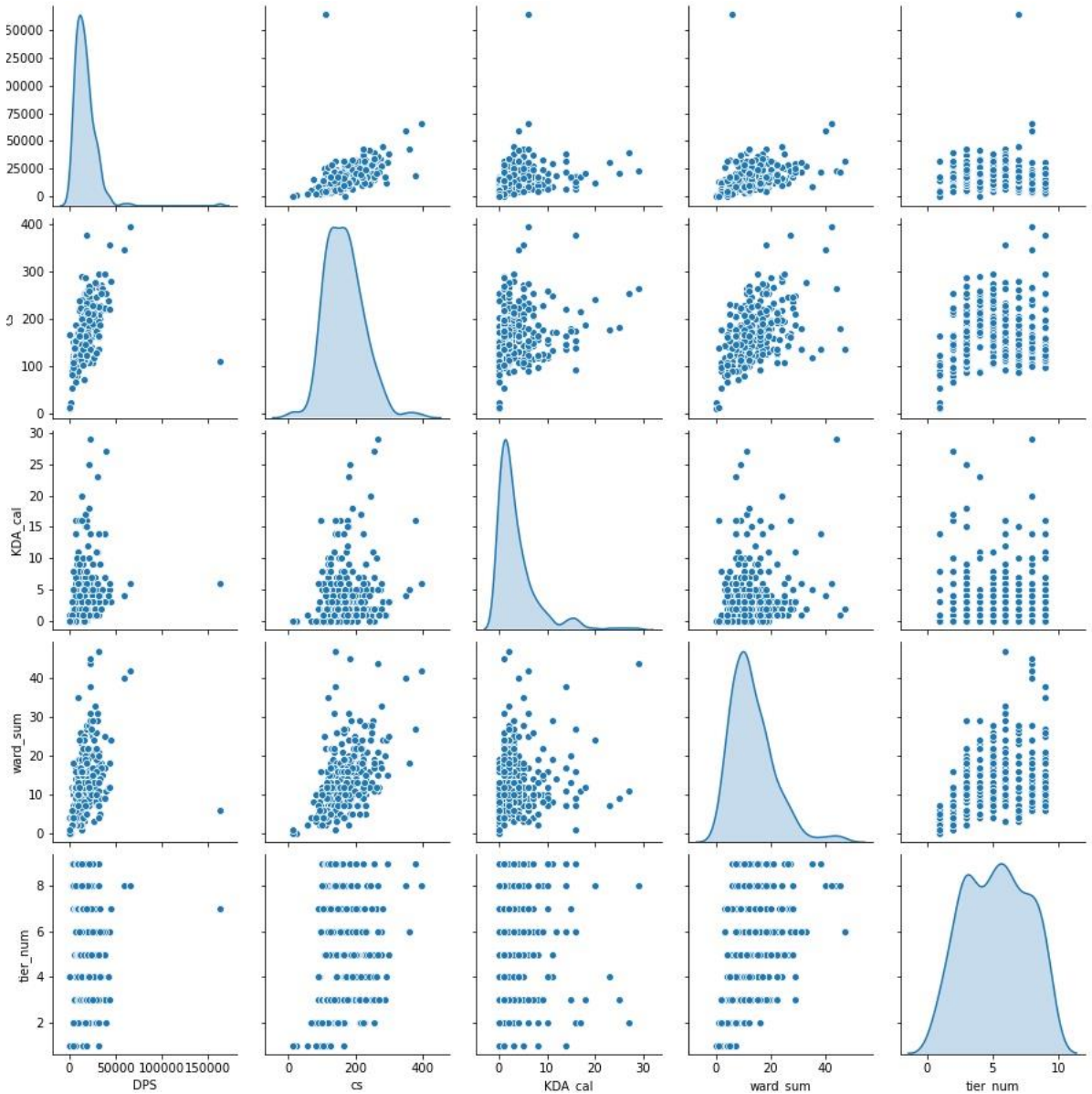
```
tier = csv['tier']
tier_num = []
for a in range(len(csv)):
    for b in range(len(tier_uni)):
        if (tier[a] == tier_uni[b]):
            tier_num.append(b+1)
```

상관계수분석




```
plt.figure(figsize=(15,15))
sns.heatmap(data = df.corr(), annot=True,
fmt = '.3f', linewidths=.5, cmap='Reds')

plt.savefig('correlation analysis.jpg')
plt.show()
```



작은 상관계수

=> 회귀를 안 쓰고
분류모델을 사용 할 것이다.

4. Make dataset

```
tier_no=csv['tier']
tier_name=[]
for i in range(len(tier_no)):
    tier=tier_no[i].split(' ')
    tier_name.append(tier[0])

tier_uni=csv['tier'].unique()
tier_uni=['Iron','Bronze','Silver','Gold','Platinum','Diamond','Master','Grandmaster','Challenger']

tier = csv['tier']
tier_num = []
for a in range(len(csv)):
    for b in range(len(tier_uni)):
        if (tier_name[a] == tier_uni[b]):
            tier_num.append(b+1)
len(tier_num)
```

분류모델을 사용하기 위해 라벨(정답)을 24개에서 10개로 줄였다.

```
df = pd.DataFrame({'DPS':DPS,'cs':cs,'cspm':cspm,'time_minute':time_minute,'KDA_K':KDA_K,'KDA_D':KDA_D,'KDA_A':KDA_A,
                   'KDA_cal':KDA_cal,'ward1':ward1,'ward2':ward2,'ward3':ward3,'ward_sum':ward_sum,
                    'DPS_pt':DPS_pt, 'tier_num':tier_num})
df
```

	DPS	cs	cspm	time_minute	KDA_K	KDA_D	KDA_A	KDA_cal	ward1	ward2	ward3	ward_sum	DPS_pt	tier_num
0	31367	296	9.3	31	6	5	11	3	0	10	5	15	1011.838710	8
1	24054	141	4.4	31	9	6	13	3	7	11	8	26	775.935484	8
2	18880	377	11.8	31	9	1	7	16	2	10	15	27	609.032258	8
3	25399	108	4.5	24	14	4	13	6	8	16	1	25	1058.291667	8
4	18421	194	8.1	24	6	3	8	4	4	7	4	15	767.541667	8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
295	54	11	0.8	14	0	1	0	0	0	0	0	0	3.857143	0
296	133	13	0.9	14	0	1	0	0	0	1	0	1	9.500000	0
297	4205	110	7.8	14	1	1	4	5	0	4	0	4	300.357143	0
298	14039	103	4.2	24	5	8	3	1	0	4	0	4	584.958333	0
299	17955	123	5.0	24	9	2	8	8	0	5	0	5	748.125000	0

```
# split train and test

train_dataset = df.sample(frac=0.8,random_state=0)
test_dataset = df.drop(train_dataset.index)
```

```
print(np.shape(train_dataset))
print(np.shape(test_dataset))
```

```
(240, 14)
(60, 14)
```

Train으로 훈련시키고 Test로 확인

정규화

```
train_labels = train_dataset.pop('tier_num')
test_labels = test_dataset.pop('tier_num')
```

```
normed_train_data =(train_dataset - train_stats['mean']) / train_stats['std']
normed_test_data = (test_dataset - test_stats['mean']) / test_stats['std']
```

```
normed_train_data
```

	DPS	cs	cspm	time_minute	KDA_K	KDA_D	KDA_A	KDA_cal	ward1	ward2	ward3	ward_sum	DPS_pt
208	1.614467	1.513372	1.948201	0.011367	3.230507	-1.272104	-0.126666	5.169211	-0.587790	0.306584	-0.604384	-0.268785	1.346767
188	-0.491609	-0.663707	-0.455907	-0.562950	-1.194845	1.331776	-1.174937	-0.856421	-0.587790	0.071504	-0.013540	-0.144011	-0.268457
12	-0.803602	-0.629150	1.085188	-1.424426	-0.973578	-0.946619	-0.965283	-0.410078	-0.587790	-0.398657	-0.308962	-0.518335	-0.404141
221	0.008583	0.856793	0.468750	0.585684	0.796563	-1.272104	0.082988	2.937496	0.660614	-0.633737	-0.308962	-0.268785	-0.112010
239	0.962172	0.217492	0.715325	-0.419371	1.239099	-1.272104	0.921605	4.276525	-1.003925	-0.398657	-0.604384	-0.767884	1.049281
...	...	...	...	...	...	...	...	...	...	...	...	...	...
11	-0.631267	-0.698263	0.900257	-1.424426	-0.088507	-0.621134	-1.174937	-0.410078	-0.171656	-0.398657	-0.604384	-0.518335	-0.174043
119	1.234138	1.858940	0.592037	1.590739	2.124169	-0.295649	-0.545974	0.259437	-0.171656	0.541664	0.872726	0.604637	0.424291
102	1.126138	0.891350	0.530394	0.585684	1.239099	0.029836	2.179530	0.259437	-1.003925	-0.398657	-0.013540	-0.518335	0.683797
35	-0.780386	-0.611871	1.270119	-1.568005	-1.194845	-0.295649	-1.384591	-0.856421	-0.171656	-0.163577	-0.899806	-0.518335	-0.330188
57	-0.764375	-1.043832	-0.517551	-0.993688	-1.194845	1.006291	-0.965283	-0.856421	-0.587790	-0.868817	-0.604384	-0.892659	-0.456878

5. Softmax Classification

```
model =keras. models.Sequential()
model.add(layers.Dense(64, activation = 'relu', input_shape = [None, 13]))
# model.add(Dropout(0.5))
# model.add(layers.Dense(64, activation = 'relu'))
model.add(layers.Dense(9, activation = 'softmax'))

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
```

티어 9개로 분류

```
model.summary()
```

```
Model: "sequential_27"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_55 (Dense)             (None, None, 64)          896
_________________________________________________________________
dense_56 (Dense)             (None, None, 9)           585
=================================================================
Total params: 1,481
Trainable params: 1,481
Non-trainable params: 0
_________________________________________________________________
```

모델링

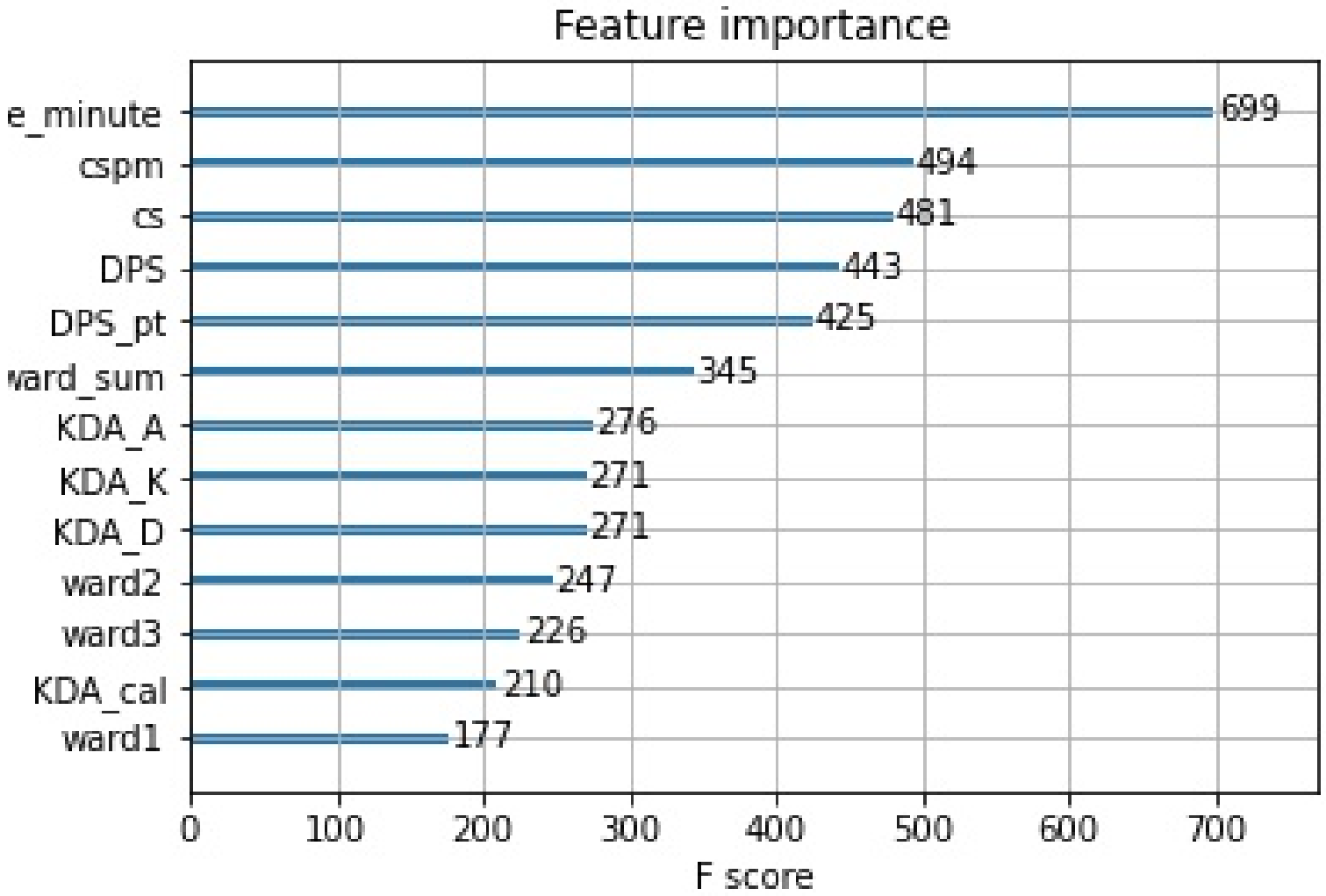
5. Softmax Classification

```
train_score = model.evaluate(normed_train_data, train_labels, verbose=0)
test_score = model.evaluate(normed_test_data, test_labels, verbose=0)

print('Train accuracy:', train_score[1])
print('Test accuracy:', test_score[1])
```

```
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for
type=float32), but it was called on an input with incompatible shape (Non
Train accuracy: 0.5208333134651184
Test accuracy: 0.3333333432674408
```

6. XGB Classifier



XGB에서
중요한 특성

```
train_pred = model1.predict(normed_train_data)
test_pred = model1.predict(normed_test_data)

train_acc = accuracy_score(train_pred, train_labels)
test_acc = accuracy_score(test_pred, test_labels)
print("Train Accuracy: %.2f%%" % (train_acc * 100.0))
print("Test Accuracy: %.2f%%" % (test_acc * 100.0))
```

```
Train Accuracy: 100.00%
Test Accuracy: 31.67%
```

결과 분석

- 한판의 데이터에 대해서는 티어 예측에 썩 도움이 되지 않는다.

- 이민수의 분석:
챌린저도 제어와드 안박는다(거의). 아이언이나 챌린저나

- 티어가 아닌 승패예측을 했으면 더 잘 되었을것 같다.

- 티어를 예측하려면 특정 사람에 대해 여러 게임의 데이터를 가져왔어야 했다.

- 포지션을 고려해야 할것같다.