PUBG Finish Placement Prediction

Competion

PUBG Finish Placement Prediction (Kernels Only)

참고 notebook

EDA for the popular battle royale game PUBG

Competion 설명

개요

PUBG는 100이 계속해서 줄어드는 전투장 속에서 최후의 1인 혹은 1팀이 남을 때 까지 전투를 하는 배틀로얄 FPS 게임이다.

이 Competion은 Player의 경기 내용에관한 데이터(kills, revives, elo 등등)를 통해 해당 Player의 등수를 예측하고 실제 등수와의 MSE가 가장 작은 모델을 만드는 것이 목표다.

Data

Train Data : 4446966 x 29

Test Data : 1934174 x 28

- **ID** : 플레이어 식별자
- **groupID** : 팀 식별자
- **matchID** : 게임 식별자
- **DBNOs** : 적을 기절시킨 횟수
- **assists** : 죽이진 않았지만 데미지를 준 적의 수
- **boosts** : 부스트 아이템 사용 횟수(진통제, 드링크, 아드레날린 주사)
- **damageDealt** : 총 넣은 데미지 량
- **headshotKills** : 헤드샷 킬 수
- **heals** : 힐 아이템 사용 수(붕대, 구급상자, 키트)
- **killPlace** : 게임 내에서 킬 등수
- **killPoints** : 게임 전체 킬 랭킹(킬만 고려한 랭킹)
- **killStreaks** : 한번의 교전에서의 최다 킬 수
- **kills** : 킬 수
- **longestKill** : 장커리 킬 거리
- **matchDuration** : 경기 시간(초)
- **matchType** : Solo, Duo, Squard

- **rankPoints** : 전체 랭킹(킬 + 승리 모두를 고려한 랭킹)
- **revives** : 동료를 살린 횟수
- **rideDistance** : 차량 이동 거리(meter)
- **roadKills** : 차로 죽인 횟수
- **swimDistance** : 수영한 거리(meter)
- **teamKills** : 팀킬 횟수
- **vehicleDestroys** : 차량 폭파 횟수
- **walkDistance** : 걸어간 거리(meter)
- **weaponsAcquired** : 획득한 무기 종류 갯수
- **winPoints** : 승리 전체 랭킹(승리를 고려한 랭킹)
- **numGroups** : 게임 내 총 그룹 수, 게임 참가 인원
- **maxPlace** : 게임내 최악의 등수, 대기실 참가 인원
- **winPlacePerc** : Target. 예측 등수를 0.0~1.0사이의 값으로 변환한 값.

정리

총 4446966 by 29의 Train Data를 통해 모델을 생성하고, 1934174 개의 test 데이터에대해 MSE값을 측정하면 된다. 분류 문제가 아닌 회귀 문제.

중점적으로 본 사항

시각화

정제되지 않은 데이터 더미들을 numpy, plot 등등 다양한 라이브러리를 이용해 시각적, 통계적으로 이해하고 분석하는 다양한 방식을 새롭게 배우고 공부해야할 필요성을 느꼈다.

Feature Engineering

시각화를 통해 완전히 이해한 데이터를 바탕으로 종속적인 속성들을 묶거나 새롭게 변형킬 수도 있다는 것을 배웠다.

시각화

numpy

> quantile()
>
> 데이터 조작(필요 범위 뽑아내거나 설정)

np.quantile() : 분위수

자료 크기 순위에 따른 위치값이다. 분위수를 통해 **"대다수"** 와 **"이상치"** 를 판별해 시각화, NA값 설정, feature engineering 등을 할 수 있다.

[in]

```
print("The average person kills {:.4f} players, \
99% of people have {} kills or less, \
while the most kills ever recorded is
{}.".format(train['kills'].mean(),train['kills'].quantile(0.99),
train['kills'].max()))
```

[out]

```
The average person kills 0.9345 players, 99% of people have 7.0 kills or less, while the most kills ever recorded is 60.
```

이 코드는 평균 킬 수, 상위 99% 사람의 킬 수, 그리고 킬 최대값을 구하는 코드이다.

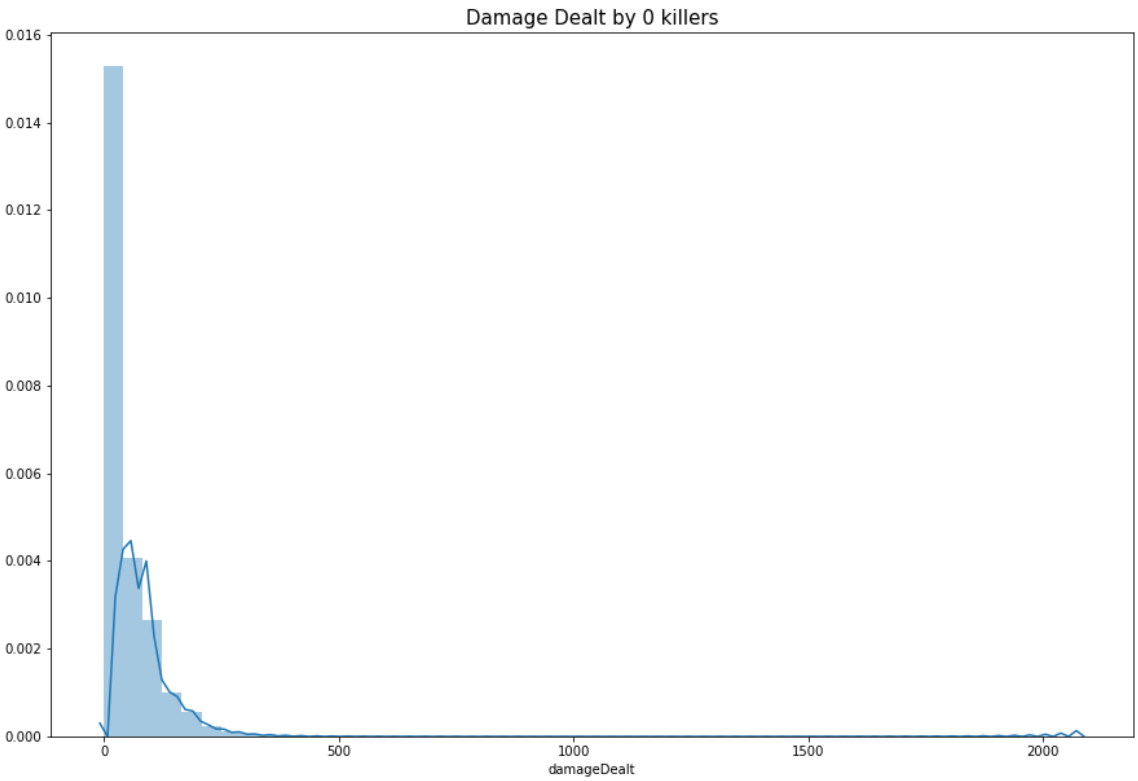
연속형 데이터의 경우에 한하여 boxplot을 간단하게 글로 표현했다고 볼 수 있다.

데이터 조작

numpy 매트릭스의 데이터들을 마치 데이터 베이스 SQL처럼 조작을 할 수 있다.

`data[-data의 조건-]` 을 이용하면 전체 데이터에서 원하는 조건을 충족시키는 데이터만 뽑아서 새로운 sub dataset을 다룰 수 있게 된다.

```
data = train.copy() #pandas.DataFrame.copy()는 default값으로 deep copy를 해준다.
data = data[data['kills']==0] # data에서 kills가 0인 부분만 골라 새로운 sub dataset 정의
plt.figure(figsize=(15,10))
plt.title("Damage Dealt by 0 killers",fontsize=15)
sns.distplot(data['damageDealt'])
plt.show()
```



이런식으로 sub data를 이용해 그래프를 그리거나, 조건에 맞는 data들의 집계를 구할 수 있다.

그래프

> loc(), astype(), sort_values()
>
> countplot()
>
> distplot()
>
> jointplot()
>
> boxplot()
>
> pointplot()
>
> subplots()
>
> heatmap()
>
> pairplot()

loc(), astype(), sort_values()

loc() : Pandas Data Frame에서 조건에 맞는 Data Frame의 행 or 열을 **조회**하고 **수정**할 수 있는 메서드다.

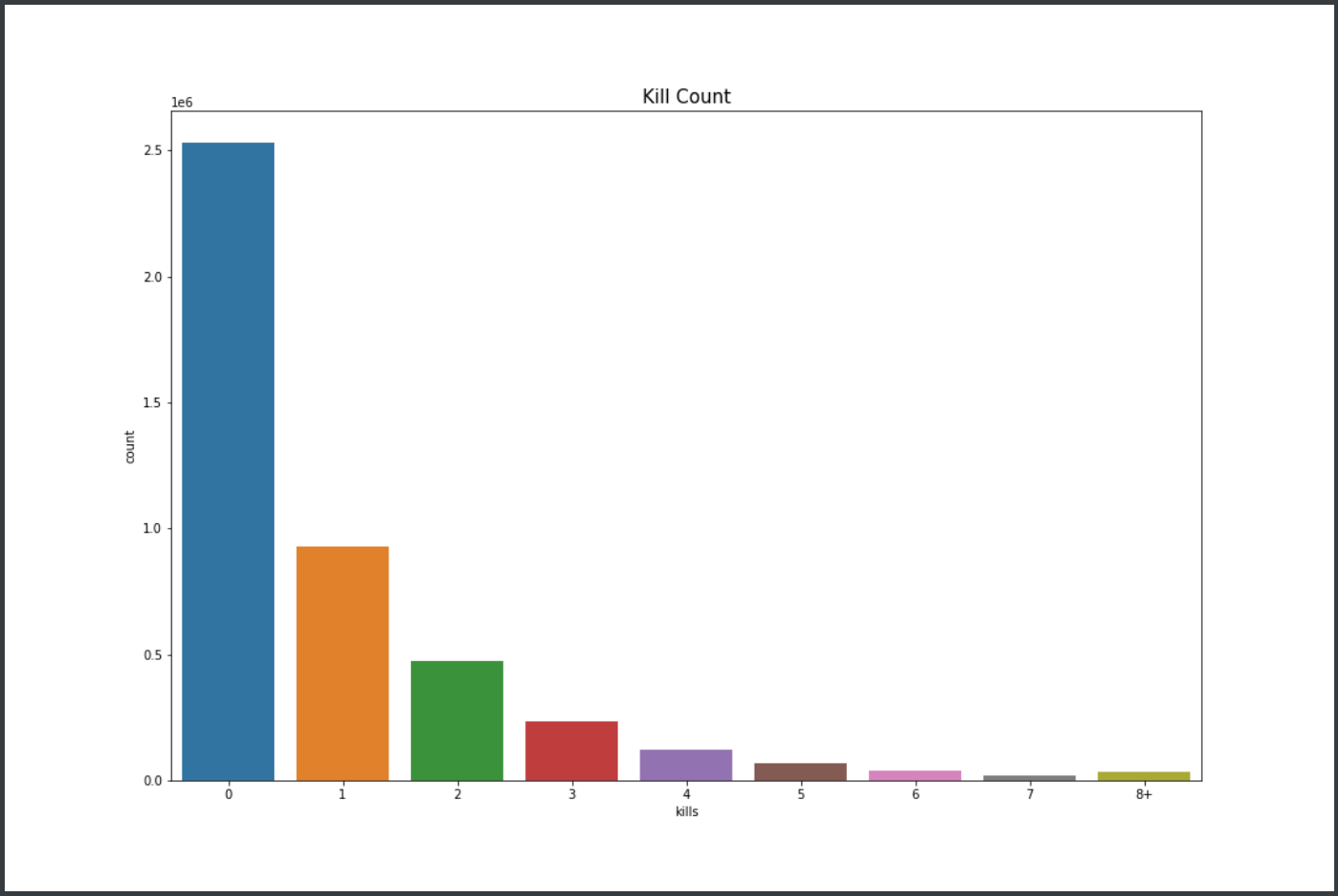
`data.loc[data['kills'] > data['kills'].quantile(0.99)] = '8+'` 는 **"data"**라는 DataFrame의 **"kills"**속성에서 상위 99% 이상인 값들을 전부 **'8+'**로 바꿔주는 코드다.

astype('타입') : 값들을 모두 지정한 타입값으로 바꿔준다.

`data['kills'].astype('str')` 는 **'kills'**의 값들을 모두 **'str'** 타입으로 바꿔준다.

sort_values() : Data Frame의 속성을 정렬 시킨다.

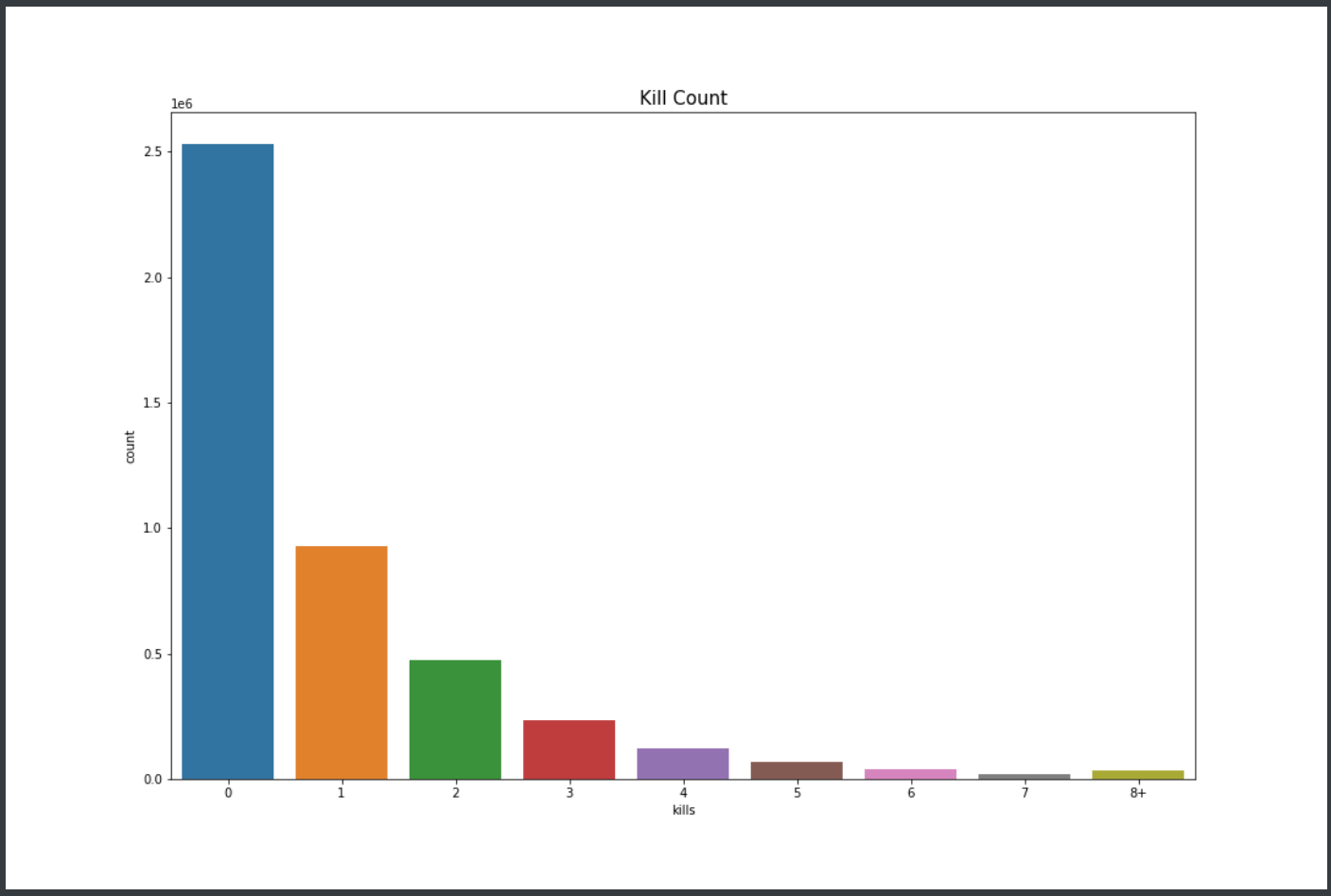
```
data = train.copy() # 깊은 복사
data.loc[data['kills'] > data['kills'].quantile(0.99)] = '8+' # 상위 99%값들을 모두 8+
str타입으로 바꿈
plt.figure(figsize=(15,10)) # figure size 설정
sns.countplot(data['kills'].astype('str').sort_values()) # kills의 타입을 str로 바꾸고 정
렬(0, 1, .. 8)
plt.title("Kill Count",fontsize=15) # figure title 지정
plt.show() # plt 출력
```



> Plot 그리는 방법들은 subplot을 제외하고 모두 비슷하기 때문에 각 plot별 특징만 알아보았다.

countplot()

data를 이산적인 막대그래프 형태로 해당하는 값의 갯수를 파악할 수 있다.

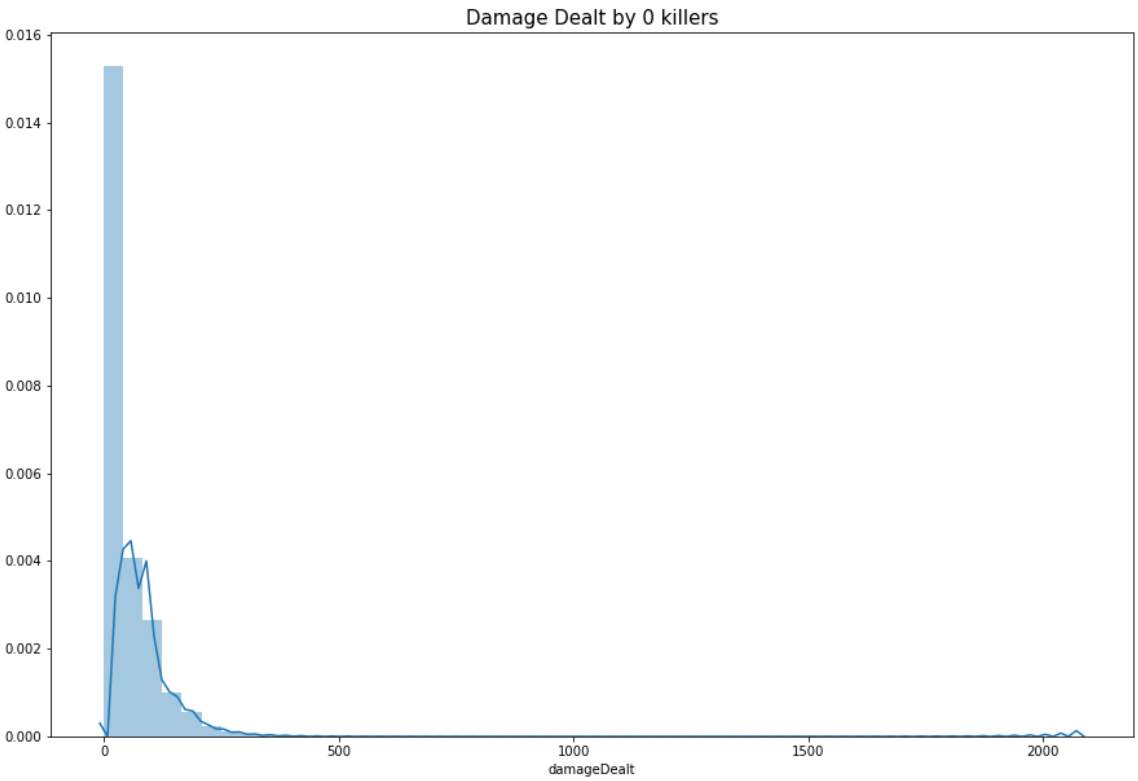


displot()

막대 그래프와 곡선 그래프가 동시에 나타나지만, 곡선 그래프가 더 중요하다.

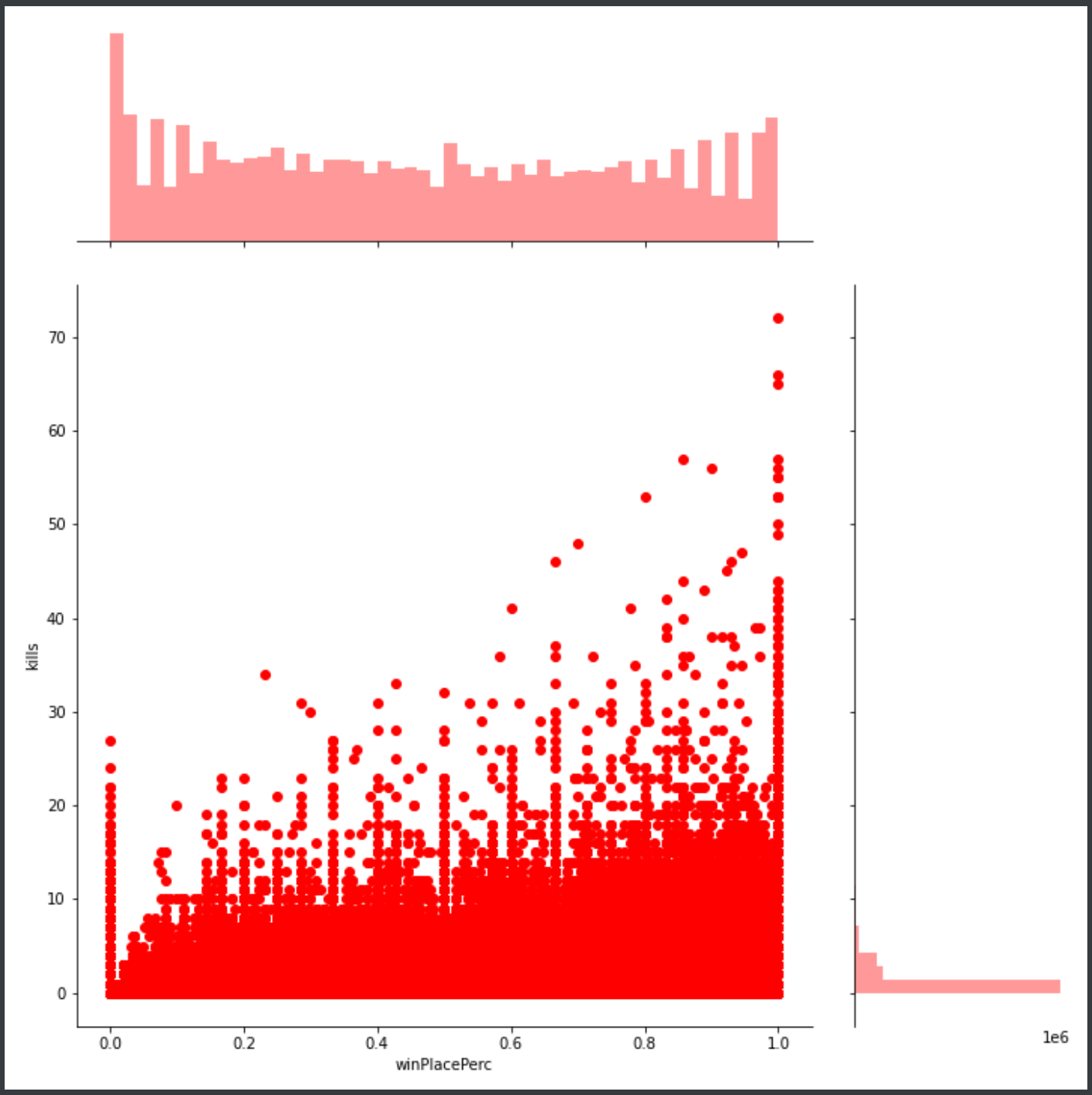
변화의 추이를 분석하기위한 그래프.

옵션으로 막대그래프를 제거할 수 있다.



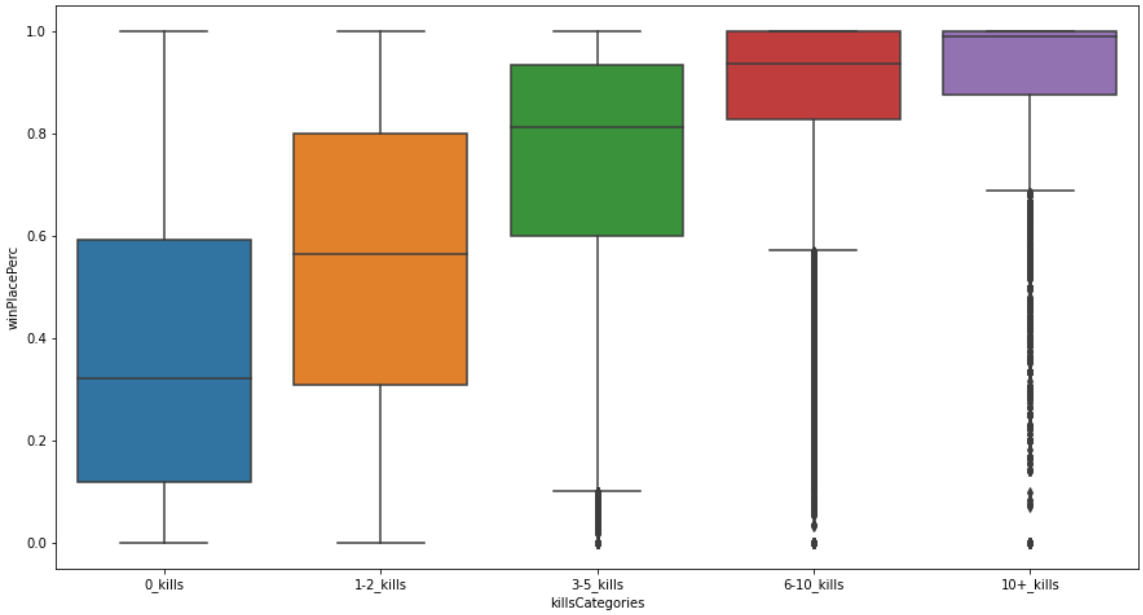
jointplot()

각 속성의 막대 그래프와 속성 2개의 scatter plot(상관도)값을 보여준다.



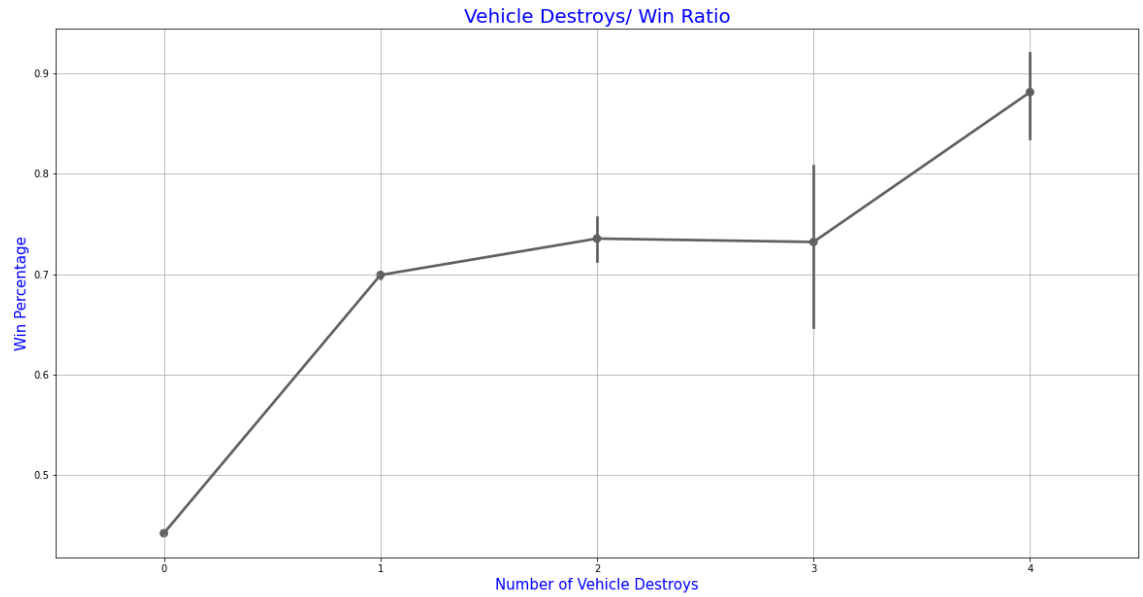
boxplot()

값의 치우쳐짐을 한눈에 알 수있는 boxplot을 보여준다.



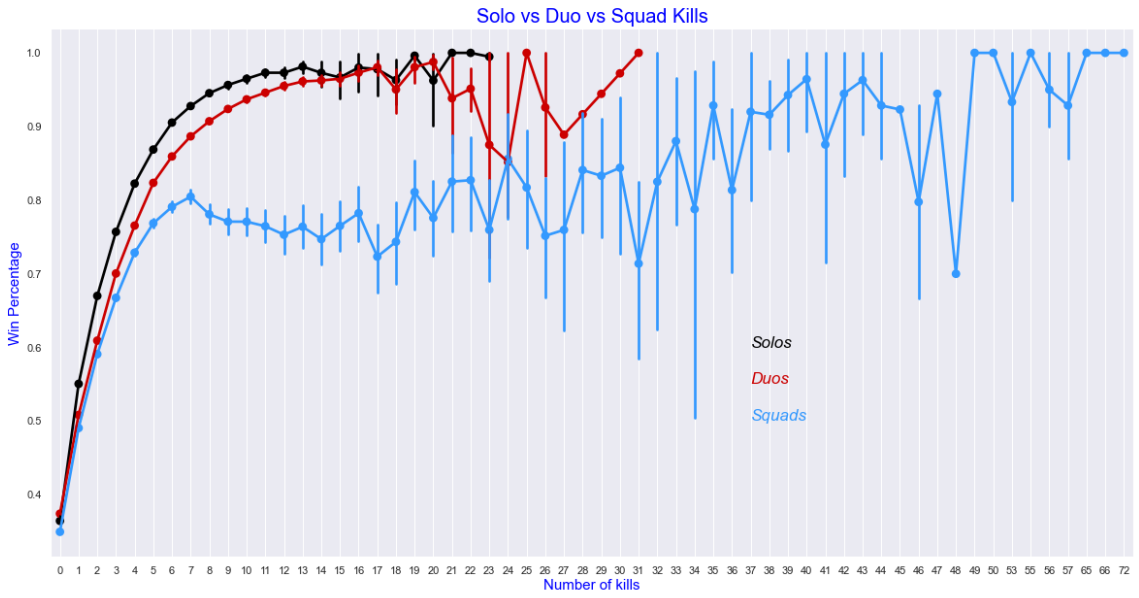
pointplot()

각 값의 평균값의 point를 이어서 그래프로 보여준다.



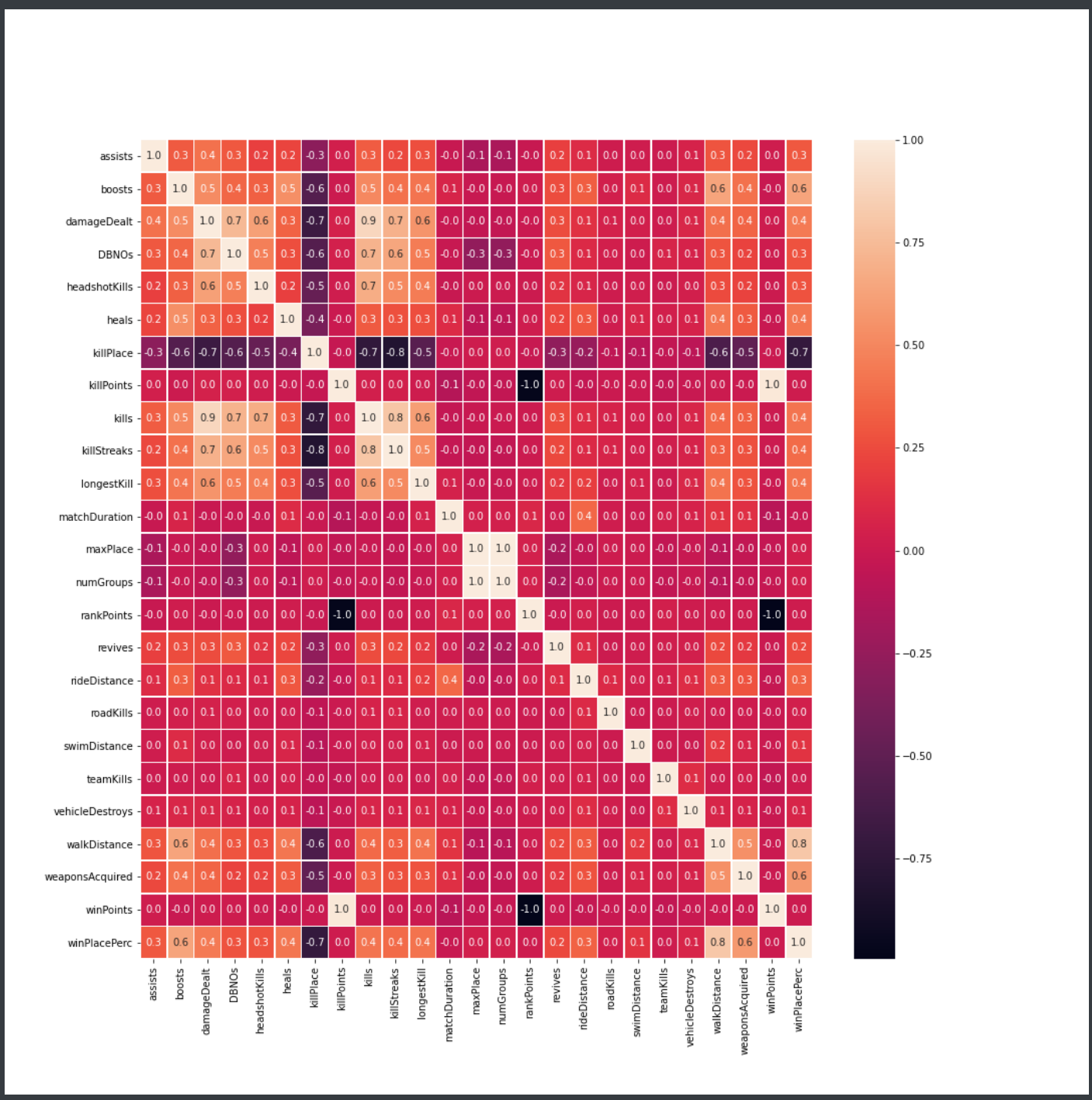
subplots()

여러개의 속성을 바둑판 형식으로 보여주는 subplot과 달리 subplots는 한 그래프안에 여러 속성을 표현할 수 있다.



heatmap()

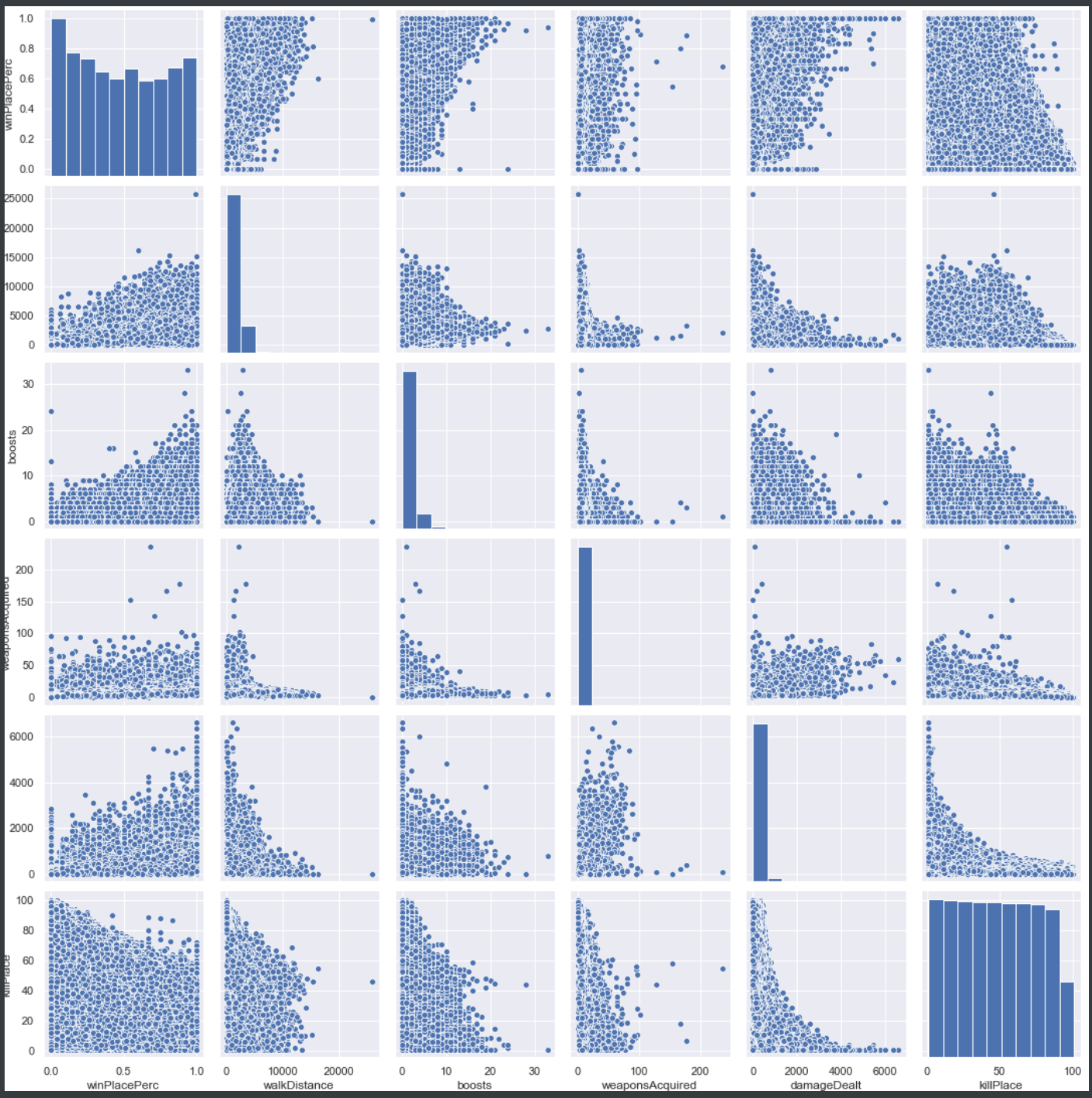
전체 속성간의 상관관계를 알 수 있다.



pairplot()

jointplot의 subplot과 비슷한 역할을 한다.

heatmap과 다르게 상관관계들을 표현한다.



Feature Engineering

가중 표준화

100명 게임에서 1킬과, 96명 게임에서 1킬은 비율상 다른것을 표준화 시킴

ex) `kill norm =  (100-players) / 100 + 1`

연관된 속성끼리 비율 통합

ex) `walkDistancePerHeals =100 * walkDistance / Heals`

등등새로운 특성을 만듬.

모델링 계획

Outlier Deal(Cheater)

Feature Engineering

Solo, Duo, Sqard 로 모델을 나누어 predict

모델 후보

- LinearRegression
- RandomForest
- LGBMRegressor
- ensemble LightGBM