pubg-eda

April 10, 2022

[1]:
```
import numpy as np
import pandas as pd

import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from lightgbm import LGBMRegressor
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from sklearn.preprocessing import StandardScaler
from gc import collect
```

```
<IPython.core.display.HTML object>
```

[2]:
```
pd.set_option('display.max_columns', 40)
```

[3]:
```
submission_function=pd.read_csv('/kaggle/input/pubg-finish-placement-prediction/
↪sample_submission_V2.csv') # example of how  summition look like
```

[4]:
```
df_test=pd.read_csv('/kaggle/input/pubg-finish-placement-prediction/test_V2.
↪csv')   # pridict the winPlacePerc
# so over goal to pridict winPlacePerc for finding best correleted feature

df=pd.read_csv('/kaggle/input/pubg-finish-placement-prediction/train_V2.csv')
```

[5]:
```
df_test.head()
```

[5]:

	Id	groupId	matchId	assists	boosts	\
0	9329eb41e215eb	676b23c24e70d6	45b576ab7daa7f	0	0	
1	639bd0dcd7bda8	430933124148dd	42a9a0b906c928	0	4	
2	63d5c8ef8dfe91	0b45f5db20ba99	87e7e4477a048e	1	0	
3	cf5b81422591d1	b7497dbdc77f4a	1b9a94f1af67f1	0	0	
4	ee6a295187ba21	6604ce20a1d230	40754a93016066	0	4	

	damageDealt	DBNOs	headshotKills	heals	killPlace	killPoints	kills	\

```
0        51.46      0              0      0         73           0      0
1       179.10      0              0      2         11           0      2
2        23.40      0              0      4         49           0      0
3        65.52      0              0      0         54           0      0
4       330.20      1              2      1          7           0      3

   killStreaks  longestKill  matchDuration  matchType  maxPlace  numGroups  \
0            0         0.00           1884  squad-fpp        28         28
1            1       361.90           1811    duo-fpp        48         47
2            0         0.00           1793  squad-fpp        28         27
3            0         0.00           1834    duo-fpp        45         44
4            1        60.06           1326  squad-fpp        28         27

   rankPoints  revives  rideDistance  roadKills  swimDistance  teamKills  \
0        1500        0           0.0          0           0.0          0
1        1503        2        4669.0          0           0.0          0
2        1565        0           0.0          0           0.0          0
3        1465        0           0.0          0           0.0          0
4        1480        1           0.0          0           0.0          0

   vehicleDestroys  walkDistance  weaponsAcquired  winPoints
0                0         588.0                1          0
1                0        2017.0                6          0
2                0         787.8                4          0
3                0        1812.0                3          0
4                0        2963.0                4          0
```

[6]:
```
df.head()
```

[6]:
```
               Id         groupId         matchId  assists  boosts  \
0  7f96b2f878858a  4d4b580de459be  a10357fd1a4a91        0       0
1  eef90569b9d03c  684d5656442f9e  aeb375fc57110c        0       0
2  1eaf90ac73de72  6a4a42c3245a74  110163d8bb94ae        1       0
3  4616d365dd2853  a930a9c79cd721  f1f1f4ef412d7e        0       0
4  315c96c26c9aac  de04010b3458dd  6dc8ff871e21e6        0       0

   damageDealt  DBNOs  headshotKills  heals  killPlace  killPoints  kills  \
0         0.00      0              0      0         60        1241      0
1        91.47      0              0      0         57           0      0
2        68.00      0              0      0         47           0      0
3        32.90      0              0      0         75           0      0
4       100.00      0              0      0         45           0      1

   killStreaks  longestKill  matchDuration  matchType  maxPlace  numGroups  \
0            0         0.00           1306  squad-fpp        28         26
1            0         0.00           1777  squad-fpp        26         25
2            0         0.00           1318        duo        50         47
```

```
3            0         0.00           1436  squad-fpp        31         30
4            1        58.53           1424   solo-fpp        97         95

   rankPoints  revives  rideDistance  roadKills  swimDistance  teamKills  \
0          -1        0        0.0000          0          0.00          0
1        1484        0        0.0045          0         11.04          0
2        1491        0        0.0000          0          0.00          0
3        1408        0        0.0000          0          0.00          0
4        1560        0        0.0000          0          0.00          0

   vehicleDestroys  walkDistance  weaponsAcquired  winPoints  winPlacePerc
0                0        244.80                1       1466        0.4444
1                0       1434.00                5          0        0.6400
2                0        161.80                2          0        0.7755
3                0        202.70                3          0        0.1667
4                0         49.75                2          0        0.1875
```

[7]:
```
df.describe()
```

[7]:
```
             assists        boosts   damageDealt         DBNOs  headshotKills  \
count  4.446966e+06  4.446966e+06  4.446966e+06  4.446966e+06   4.446966e+06
mean   2.338149e-01  1.106908e+00  1.307171e+02  6.578755e-01   2.268196e-01
std    5.885731e-01  1.715794e+00  1.707806e+02  1.145743e+00   6.021553e-01
min    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00   0.000000e+00
25%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00   0.000000e+00
50%    0.000000e+00  0.000000e+00  8.424000e+01  0.000000e+00   0.000000e+00
75%    0.000000e+00  2.000000e+00  1.860000e+02  1.000000e+00   0.000000e+00
max    2.200000e+01  3.300000e+01  6.616000e+03  5.300000e+01   6.400000e+01

              heals     killPlace    killPoints         kills   killStreaks  \
count  4.446966e+06  4.446966e+06  4.446966e+06  4.446966e+06  4.446966e+06
mean   1.370147e+00  4.759935e+01  5.050060e+02  9.247833e-01  5.439551e-01
std    2.679982e+00  2.746294e+01  6.275049e+02  1.558445e+00  7.109721e-01
min    0.000000e+00  1.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
25%    0.000000e+00  2.400000e+01  0.000000e+00  0.000000e+00  0.000000e+00
50%    0.000000e+00  4.700000e+01  0.000000e+00  0.000000e+00  0.000000e+00
75%    2.000000e+00  7.100000e+01  1.172000e+03  1.000000e+00  1.000000e+00
max    8.000000e+01  1.010000e+02  2.170000e+03  7.200000e+01  2.000000e+01

        longestKill  matchDuration      maxPlace     numGroups    rankPoints  \
count  4.446966e+06   4.446966e+06  4.446966e+06  4.446966e+06  4.446966e+06
mean   2.299759e+01   1.579506e+03  4.450467e+01  4.300759e+01  8.920105e+02
std    5.097262e+01   2.587399e+02  2.382811e+01  2.328949e+01  7.366478e+02
min    0.000000e+00   9.000000e+00  1.000000e+00  1.000000e+00 -1.000000e+00
25%    0.000000e+00   1.367000e+03  2.800000e+01  2.700000e+01 -1.000000e+00
50%    0.000000e+00   1.438000e+03  3.000000e+01  3.000000e+01  1.443000e+03
75%    2.132000e+01   1.851000e+03  4.900000e+01  4.700000e+01  1.500000e+03
```

```
max    1.094000e+03   2.237000e+03  1.000000e+02  1.000000e+02  5.910000e+03

            revives  rideDistance     roadKills  swimDistance     teamKills  \
count  4.446966e+06  4.446966e+06  4.446966e+06  4.446966e+06  4.446966e+06
mean   1.646590e-01  6.061157e+02  3.496091e-03  4.509322e+00  2.386841e-02
std    4.721671e-01  1.498344e+03  7.337297e-02  3.050220e+01  1.673935e-01
min    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
25%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
50%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
75%    0.000000e+00  1.909750e-01  0.000000e+00  0.000000e+00  0.000000e+00
max    3.900000e+01  4.071000e+04  1.800000e+01  3.823000e+03  1.200000e+01

       vehicleDestroys  walkDistance  weaponsAcquired     winPoints  \
count     4.446966e+06  4.446966e+06     4.446966e+06  4.446966e+06
mean      7.918208e-03  1.154218e+03     3.660488e+00  6.064601e+02
std       9.261157e-02  1.183497e+03     2.456544e+00  7.397004e+02
min       0.000000e+00  0.000000e+00     0.000000e+00  0.000000e+00
25%       0.000000e+00  1.551000e+02     2.000000e+00  0.000000e+00
50%       0.000000e+00  6.856000e+02     3.000000e+00  0.000000e+00
75%       0.000000e+00  1.976000e+03     5.000000e+00  1.495000e+03
max       5.000000e+00  2.578000e+04     2.360000e+02  2.013000e+03

       winPlacePerc
count  4.446965e+06
mean   4.728216e-01
std    3.074050e-01
min    0.000000e+00
25%    2.000000e-01
50%    4.583000e-01
75%    7.407000e-01
max    1.000000e+00
```

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4446966 entries, 0 to 4446965
Data columns (total 29 columns):
 #   Column          Dtype
---  ------          -----
 0   Id              object
 1   groupId         object
 2   matchId         object
 3   assists         int64
 4   boosts          int64
 5   damageDealt     float64
 6   DBNOs           int64
 7   headshotKills   int64
```

```
 8   heals            int64
 9   killPlace        int64
 10  killPoints       int64
 11  kills            int64
 12  killStreaks      int64
 13  longestKill      float64
 14  matchDuration    int64
 15  matchType        object
 16  maxPlace         int64
 17  numGroups        int64
 18  rankPoints       int64
 19  revives          int64
 20  rideDistance     float64
 21  roadKills        int64
 22  swimDistance     float64
 23  teamKills        int64
 24  vehicleDestroys  int64
 25  walkDistance     float64
 26  weaponsAcquired  int64
 27  winPoints        int64
 28  winPlacePerc     float64
dtypes: float64(6), int64(19), object(4)
memory usage: 983.9+ MB
```

44 lakhs rows &

so, we have 30 columns and over goal is predicting `winPlacePerc` 1) univerent analysis for one variable

0.1 Univarient Analysis

[9]:
```
df.columns
```

[9]:
```
Index(['Id', 'groupId', 'matchId', 'assists', 'boosts', 'damageDealt', 'DBNOs',
       'headshotKills', 'heals', 'killPlace', 'killPoints', 'kills',
       'killStreaks', 'longestKill', 'matchDuration', 'matchType', 'maxPlace',
       'numGroups', 'rankPoints', 'revives', 'rideDistance', 'roadKills',
       'swimDistance', 'teamKills', 'vehicleDestroys', 'walkDistance',
       'weaponsAcquired', 'winPoints', 'winPlacePerc'],
      dtype='object')
```

[10]:
```
df.head(5)
```

[10]:

	Id	groupId	matchId	assists	boosts	\
0	7f96b2f878858a	4d4b580de459be	a10357fd1a4a91	0	0	
1	eef90569b9d03c	684d5656442f9e	aeb375fc57110c	0	0	
2	1eaf90ac73de72	6a4a42c3245a74	110163d8bb94ae	1	0	
3	4616d365dd2853	a930a9c79cd721	f1f1f4ef412d7e	0	0	

```
4  315c96c26c9aac  de04010b3458dd  6dc8ff871e21e6        0       0

   damageDealt  DBNOs  headshotKills  heals  killPlace  killPoints  kills  \
0         0.00      0              0      0         60        1241      0
1        91.47      0              0      0         57           0      0
2        68.00      0              0      0         47           0      0
3        32.90      0              0      0         75           0      0
4       100.00      0              0      0         45           0      1

   killStreaks  longestKill  matchDuration  matchType  maxPlace  numGroups  \
0            0         0.00           1306  squad-fpp        28         26
1            0         0.00           1777  squad-fpp        26         25
2            0         0.00           1318        duo        50         47
3            0         0.00           1436  squad-fpp        31         30
4            1        58.53           1424   solo-fpp        97         95

   rankPoints  revives  rideDistance  roadKills  swimDistance  teamKills  \
0          -1        0        0.0000          0          0.00          0
1        1484        0        0.0045          0         11.04          0
2        1491        0        0.0000          0          0.00          0
3        1408        0        0.0000          0          0.00          0
4        1560        0        0.0000          0          0.00          0

   vehicleDestroys  walkDistance  weaponsAcquired  winPoints  winPlacePerc
0                0        244.80                1       1466        0.4444
1                0       1434.00                5          0        0.6400
2                0        161.80                2          0        0.7755
3                0        202.70                3          0        0.1667
4                0         49.75                2          0        0.1875
```

[11]:
```
all_new_player=df['Id'].nunique()
print(f'{all_new_player} there no player for playing again')
```

```
4446966 there no player for playing again
```

[12]:
```
df.columns
```

[12]:
```
Index(['Id', 'groupId', 'matchId', 'assists', 'boosts', 'damageDealt', 'DBNOs',
       'headshotKills', 'heals', 'killPlace', 'killPoints', 'kills',
       'killStreaks', 'longestKill', 'matchDuration', 'matchType', 'maxPlace',
       'numGroups', 'rankPoints', 'revives', 'rideDistance', 'roadKills',
       'swimDistance', 'teamKills', 'vehicleDestroys', 'walkDistance',
       'weaponsAcquired', 'winPoints', 'winPlacePerc'],
      dtype='object')
```

[13]:
```
print(f'{df["Id"][0]} player id is alphnumerice & dtype {df["Id"].dtype}')
```

```
7f96b2f878858a player id is alphnumerice & dtype object
```

[14]:
```
df.shape
```

[14]:
```
(4446966, 29)
```

[15]:
```
print(f"{df['groupId'].count()} all player are different group ")
```

```
4446966 all player are different group
```

[16]:
```
print(f'{df["groupId"][0]} group id is alphnumerice & dtype {df["groupId"].
 ↪dtype} ')
```

```
4d4b580de459be group id is alphnumerice & dtype object
```

[17]:
```
print(f" we have number of distnic values {df['matchType'].nunique()} \n & also␣
 ↪distnice columns-->\n {df['matchType'].unique()} ")
```

```
 we have number of distnic values 16
 & also distnice columns-->
 ['squad-fpp' 'duo' 'solo-fpp' 'squad' 'duo-fpp' 'solo' 'normal-squad-fpp'
 'crashfpp' 'flaretpp' 'normal-solo-fpp' 'flarefpp' 'normal-duo-fpp'
 'normal-duo' 'normal-squad' 'crashtpp' 'normal-solo']
```

converting Cetgorigal values into integer range `0 to 15` in `matchType`

[18]:
```
df['matchType'].replace(df['matchType'].unique(),list(range(0,16)),inplace=True)
```

reducing size of data

[19]:
```
for column in df.columns[3:]:
    if df[column].dtype==np.int64:
        df[column]=df[column].astype(np.int8)
#     print(i)
    else:
        df[column]=df[column].astype(np.float16)
```

[20]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4446966 entries, 0 to 4446965
Data columns (total 29 columns):
 #   Column          Dtype
---  ------          -----
 0   Id              object
 1   groupId         object
 2   matchId         object
 3   assists         int8
 4   boosts          int8
 5   damageDealt     float16
 6   DBNOs           int8
```

```
 7   headshotKills    int8
 8   heals            int8
 9   killPlace        int8
 10  killPoints       int8
 11  kills            int8
 12  killStreaks      int8
 13  longestKill      float16
 14  matchDuration    int8
 15  matchType        int8
 16  maxPlace         int8
 17  numGroups        int8
 18  rankPoints       int8
 19  revives          int8
 20  rideDistance     float16
 21  roadKills        int8
 22  swimDistance     float16
 23  teamKills        int8
 24  vehicleDestroys  int8
 25  walkDistance     float16
 26  weaponsAcquired  int8
 27  winPoints        int8
 28  winPlacePerc     float16
dtypes: float16(6), int8(20), object(3)
memory usage: 237.5+ MB
```

[21]:
```
# Now, we had new data set numerical values
df.head(5)
```

[21]:

	Id	groupId	matchId	assists	boosts	\
0	7f96b2f878858a	4d4b580de459be	a10357fd1a4a91	0	0	
1	eef90569b9d03c	684d5656442f9e	aeb375fc57110c	0	0	
2	1eaf90ac73de72	6a4a42c3245a74	110163d8bb94ae	1	0	
3	4616d365dd2853	a930a9c79cd721	f1f1f4ef412d7e	0	0	
4	315c96c26c9aac	de04010b3458dd	6dc8ff871e21e6	0	0	

	damageDealt	DBNOs	headshotKills	heals	killPlace	killPoints	kills	\
0	0.00000	0	0	0	60	-39	0	
1	91.50000	0	0	0	57	0	0	
2	68.00000	0	0	0	47	0	0	
3	32.90625	0	0	0	75	0	0	
4	100.00000	0	0	0	45	0	1	

	killStreaks	longestKill	matchDuration	matchType	maxPlace	numGroups	\
0	0	0.00000	26	0	28	26	
1	0	0.00000	-15	0	26	25	
2	0	0.00000	38	1	50	47	
3	0	0.00000	-100	0	31	30	

```
4            1     58.53125           -112          2        97         95

   rankPoints  revives  rideDistance  roadKills  swimDistance  teamKills  \
0          -1        0      0.000000          0      0.000000          0
1         -52        0      0.004501          0     11.039062          0
2         -45        0      0.000000          0      0.000000          0
3        -128        0      0.000000          0      0.000000          0
4          24        0      0.000000          0      0.000000          0

   vehicleDestroys  walkDistance  weaponsAcquired  winPoints  winPlacePerc
0                0        244.75                1        -70      0.444336
1                0       1434.00                5          0      0.640137
2                0        161.75                2          0      0.775391
3                0        202.75                3          0      0.166748
4                0         49.75                2          0      0.187500
```

[22]:
```
df.isnull().sum()
```

[22]:
```
Id                0
groupId           0
matchId           0
assists           0
boosts            0
damageDealt       0
DBNOs             0
headshotKills     0
heals             0
killPlace         0
killPoints        0
kills             0
killStreaks       0
longestKill       0
matchDuration     0
matchType         0
maxPlace          0
numGroups         0
rankPoints        0
revives           0
rideDistance      0
roadKills         0
swimDistance      0
teamKills         0
vehicleDestroys   0
walkDistance      0
weaponsAcquired   0
winPoints         0
winPlacePerc      1
```

```
dtype: int64
```

we had one null values, let's drop it;

[23]:
```
df=df.dropna(axis=0) # after this process we don't have any null values in over␣
↪Data set
```

let's first see which Variable have how many unique data

[24]:
```
# df['assists'].nunique()
print('this columns have number of unique values')
for column in df.columns:
    print(column ,'--> ',df[column].nunique())
```

```
this columns have number of unique values
Id -->  4446965
groupId -->  2026744
matchId -->  47964
assists -->  20
boosts -->  27
damageDealt -->  12925
DBNOs -->  39
headshotKills -->  34
heals -->  63
killPlace -->  101
killPoints -->  256
kills -->  58
killStreaks -->  18
longestKill -->  11155
matchDuration -->  256
matchType -->  16
maxPlace -->  99
numGroups -->  100
rankPoints -->  256
revives -->  25
rideDistance -->  15734
roadKills -->  14
swimDistance -->  12589
teamKills -->  11
vehicleDestroys -->  6
walkDistance -->  15940
weaponsAcquired -->  97
winPoints -->  256
winPlacePerc -->  2268
```

let's pick randomly a variable

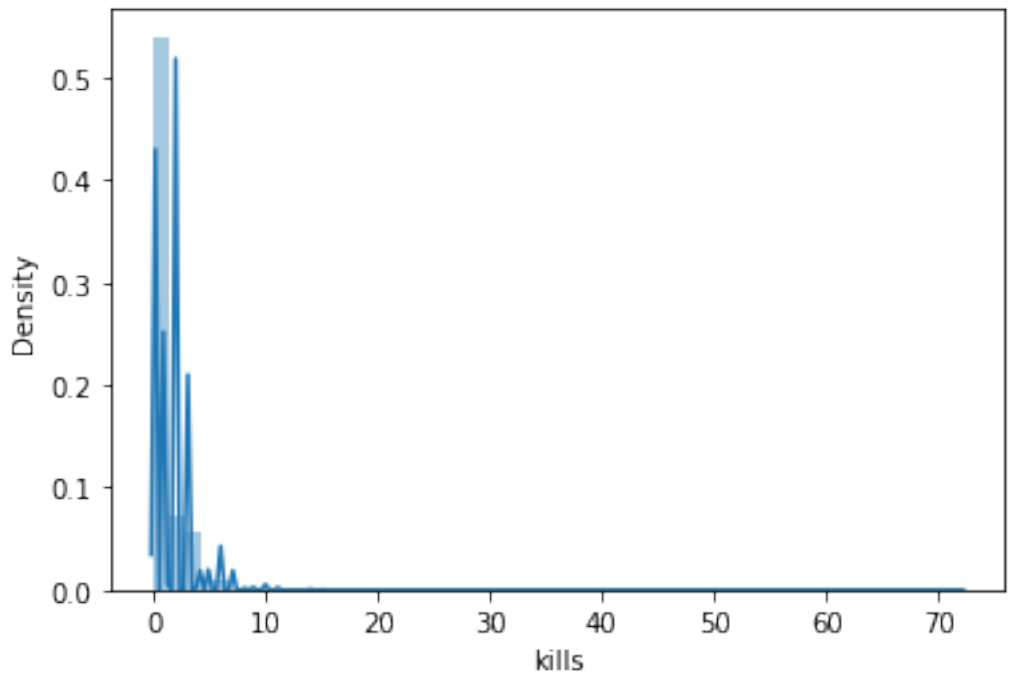
[25]:
```
df['kills'].describe()
```

[25]:
```
count    4.446965e+06
mean     9.247835e-01
std      1.558445e+00
min      0.000000e+00
25%      0.000000e+00
50%      0.000000e+00
75%      1.000000e+00
max      7.200000e+01
Name: kills, dtype: float64
```

[26]:
```
sns.distplot(df['kills'])
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[26]:
```
<AxesSubplot:xlabel='kills', ylabel='Density'>
```



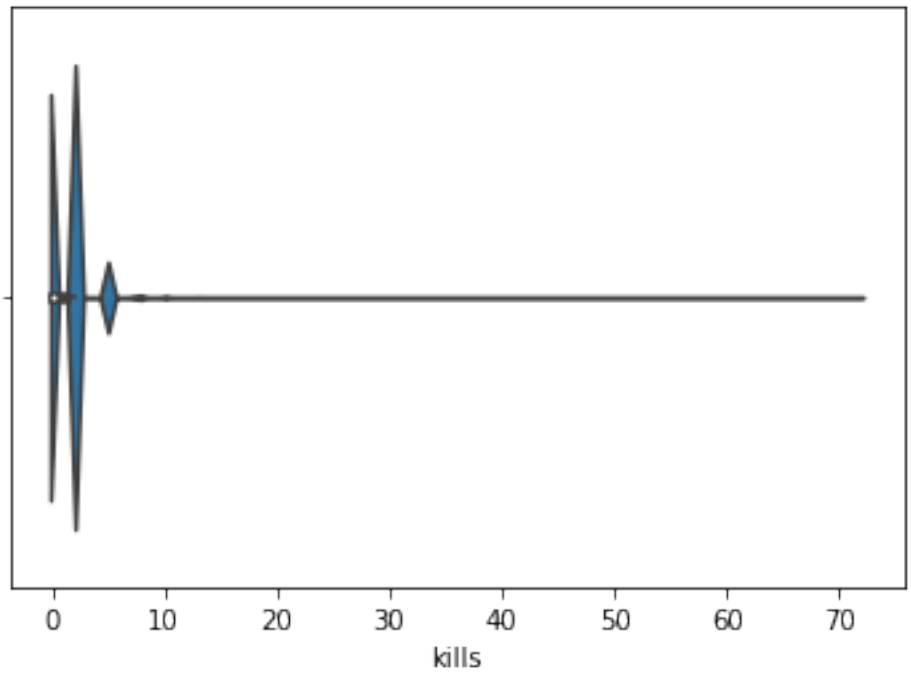
[27]:
```
sns.violinplot(df['kills'],)
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
```

```
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_19/1797699757.py in <module>
      1 sns.violinplot(df['kills'],)
----> 2 plt.show()

NameError: name 'plt' is not defined
```



[ ]:
```
sns.jointplot(df['kills'])
```

[ ]:
```
sns.heatmap(df.corr())
```

[ ]:
```
df_corr=df.corr()
```

[ ]:
```
sns.heatmap(df_corr[df_corr>0.5])
```

[ ]:
```
df_corr[df_corr>0.5]  # now we have only correlation between 0-1
```

[ ]:
```
from gc import collect
collect()
```

[ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

[ ]:
```
plt.scatter(data=df,x='kills',y='winPlacePerc')
plt.show()
```

[ ]:
```

```

0.2 Bivarient Analysis

0.3 Multivarient Analysis

[ ]:
```

```

0.4 Modeling

[ ]:
```
x=df.drop(['Id', 'groupId', 'matchId','winPlacePerc'],axis=1)
y=df['winPlacePerc']
```

0.4.1 Data preprocessing

[ ]:
```
scale_one_range=StandardScaler()
after_scale_df=scale_one_range.fit_transform(x)
```

0.5 Data spliting

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(
    after_scale_df, y, test_size=0.10, random_state=42)
```

linearRegression

[ ]:
```
model_li=LinearRegression()
model_li.fit(X_train,y_train)
# for bias score
exit_Train_data_x_y_predict=model_li.predict(X_train)
print(f' this bias score␣
 ↪{mean_absolute_error(y_train,exit_Train_data_x_y_predict)}')
## for varienc socre
y_test_predict=model_li.predict(X_test)
print(f' this bias score {mean_absolute_error(y_test,y_test_predict)}')
```

[ ]:
```
from sklearn.linear_model import Ridge,Lasso
model_rg=Ridge(alpha=00.1)
```

```
model_rg.fit(X_train,y_train)
# for bias score
exit_Train_data_x_y_predict=model_rg.predict(X_train)
print(f' this bias score␣
 ↪{mean_absolute_error(y_train,exit_Train_data_x_y_predict)}')
## for varienc socre
y_test_predict=model_rg.predict(X_test)
print(f' this bias score {mean_absolute_error(y_test,y_test_predict)}')
```

0.6 lasso

[ ]:
```
model_lass=Lasso()
model_lass.fit(X_train,y_train)
# for bias score
exit_Train_data_x_y_predict=model_lass.predict(X_train)
print(f' this bias score␣
 ↪{mean_absolute_error(y_train,exit_Train_data_x_y_predict)}')
## for varienc socre
y_test_predict=model_lass.predict(X_test)
print(f' this bias score {mean_absolute_error(y_test,y_test_predict)}')
```

[ ]:
```
# summision round
model_lass.fit(after_scale_df,y)
```

[ ]:
```

```

[ ]:
```
x=df_test.drop(['Id', 'groupId', 'matchId',],axis=1)
x['matchType'].replace(x['matchType'].unique(),list(range(0,16)),inplace=True)
after_scale_df_test=scale_one_range.fit_transform(x)
```

[ ]:
```
y_predict=model_lass.predict(after_scale_df_test)
```

[ ]:
```
submission_function['winPlacePerc']=y_predict
```

[ ]:
```
submission_function
```

[ ]:
```
submission_function.to_csv('submission.csv')
```

[ ]:
```
from sklearn.svm import SVR
```

[ ]:
```
model_svr=SVR()
model_svr.fit(X_train,y_train)
# for bias score
exit_Train_data_x_y_predict=model_svr.predict(X_train)
print(f' this bias score␣
 ↪{mean_absolute_error(y_train,exit_Train_data_x_y_predict)}')
```

```
## for varienc socre
y_test_predict=model_svr.predict(X_test)
print(f' this bias score {mean_absolute_error(y_test,y_test_predict)}')
```

[ ]:
```
from sklearn.neighbors import KNeighborsRegressor

neigh = KNeighborsRegressor(n_neighbors=2)

neigh.fit(X_train,y_train)
# for bias score
exit_Train_data_x_y_predict=neigh.predict(X_train)
print(f' this bias score␣
 ↪{mean_absolute_error(y_train,exit_Train_data_x_y_predict)}')
## for varienc socre
y_test_predict=neigh.predict(X_test)
print(f' this bias score {mean_absolute_error(y_test,y_test_predict)}')
```

[ ]:
```
from sklearn.preprocessing import PolynomialFeatures   # this cell need all␣
 ↪data for changing polynomial feature left on process heree.
model_pol=PolynomialFeatures(2)
polinomila_data=model_pol.fit_transform(X_train)
# for bias score
model_li=LinearRegression()
model_li.fit(polinomila_data,y_train)
# for bias score
exit_Train_data_x_y_predict=model_li.predict(X_train)
print(f' this bias score␣
 ↪{mean_absolute_error(y_train,exit_Train_data_x_y_predict)}')
## for varienc socre
y_test_predict=model_li.predict(X_test)
print(f' this bias score {mean_absolute_error(y_test,y_test_predict)}')
```

decision tree regression

[ ]:
```
model_de=DecisionTreeRegressor()
model_de.fit(X_train,y_train)
# for bias score
exit_Train_data_x_y_predict=model_de.predict(X_train)
print(f' this bias score␣
 ↪{mean_absolute_error(y_train,exit_Train_data_x_y_predict)}')
## for varienc socre
y_test_predict=model_de.predict(X_test)
print(f' this varience score {mean_absolute_error(y_test,y_test_predict)}')
```

0.6.1 Essamble Methods

Randomforestregression

```
[ ]: model_re=RandomForestRegressor(n_estimators=4)
     model_re.fit(X_train,y_train)
     # for bias score
     exit_Train_data_x_y_predict=model_re.predict(X_train)
     print(f' this bias score␣
      ↪{mean_absolute_error(y_train,exit_Train_data_x_y_predict)}')
     ## for varienc socre
     y_test_predict=model_re.predict(X_test)
     print(f' this varience score {mean_absolute_error(y_test,y_test_predict)}')
```

```
[ ]: collect()
```

```
[ ]: from sklearn.ensemble import ␣
      ↪AdaBoostRegressor,BaggingRegressor,ExtraTreesRegressor,GradientBoostingRegressor,HistGradie
```

```
[ ]: model_list=[]
     for models in␣
      ↪[AdaBoostRegressor,BaggingRegressor,ExtraTreesRegressor,GradientBoostingRegressor,HistGradi
      ↪
        model_esamble=models()
        print(model_esamble)
        model_esamble.fit(X_train,y_train)
        # for bias score
        exit_Train_data_x_y_predict=model_esamble.predict(X_train)
        print(f' this bias score␣
      ↪{mean_absolute_error(y_train,exit_Train_data_x_y_predict)}')
        ## for varienc socre
        y_test_predict=model_esamble.predict(X_test)
        print(f' this varience score {mean_absolute_error(y_test,y_test_predict)}')
        model_list.append(model_esamble)
        collect()
        print('\n','-'*15)
```

lgbmregressor

```
[ ]:  lgbm = LGBMRegressor(
            num_leaves=2 ** np.random.randint(3, 8),
            learning_rate = 10 ** (-np.random.uniform(0.1,2)),
            n_estimators = 100,
            min_child_samples = 1000,
            subsample=np.random.uniform(0.5,1.0),
            subsample_freq=1,
            n_jobs= -1
        )
     lgbm.fit(X_train,y_train,eval_set = (X_train, y_test), early_stopping_rounds =␣
      ↪10)
```

```
exit_Train_data_x_y_predict=lgbm.predict(X_train)
print(f' this bias score {mean_absolute_error(y_train,exit_data_x_y_predict)}')
## for varienc socre
y_test_predict=lgbm.predict(X_test)
print(f' this bias score {mean_absolute_error(y_test,y_test_predict)}')

collect()
```

[ ]:
```
model_x = xgb.XGBRegressor(
    n_estimators=800,
    learning_rate=0.03,
    max_depth=12,
    subsample=0.95,
    colsample_bytree=0.9,
    #colsample_bylevel=0.75,
    missing=-999,
    random_state=1,
    tree_method='gpu_hist'
    )
model_x.(X_train,y_train,eval_set = (X_train, y_test), early_stopping_rounds =␣
 ↪10)

exit_Train_data_x_y_predict=model_x.predict(X_train)
print(f' this bias score {mean_absolute_error(y_train,exit_data_x_y_predict)}')
## for varienc socre
y_test_predict=model_x.predict(X_test)
print(f' this bias score {mean_absolute_error(y_test,y_test_predict)}')

collect()
```

adboost

[ ]:
```

```

DNN

[ ]:
```
X_train, X_test, y_train, y_test
```

[ ]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import ModelCheckpoint
NN_model = Sequential()
NN_model.add(Dense(x_train.shape[1],  input_dim = X_train.shape[1],␣
 ↪activation='relu'))
NN_model.add(Dense(136, activation='relu'))
NN_model.add(Dense(136, activation='relu'))
NN_model.add(Dense(136, activation='relu'))
NN_model.add(Dense(136, activation='relu'))
```

```
# output Layer
NN_model.add(Dense(1, activation='linear'))

# Compile the network :
NN_model.compile(loss='mean_absolute_error', optimizer='adam',␣
 ↪metrics=['mean_absolute_error'])

NN_model.fit(x=X_train,
                 y=y_train,
                 batch_size=1000,
                 epochs=30, )
```