```
import pandas as pd
import numpy as np
from pandas.plotting import scatter_matrix
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
import seaborn as sns
sns.set(color_codes=True)
import matplotlib.pyplot as plt


# Specify the location of the dataset.
MLB = 'NewPitchingVolations.csv'


# Load the data into a Pandas DataFrame
df= pd.read_csv (MLB, header=None)


df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	13	14	15	16	17	18	19
0	Jose	A. Ferrer	WSN	4.100	63	29	34	0.000	0.310	0.190	...	3.430	0.070	0.100	0.300	18.900	0.500	72.000
1	Fernando	Abad	COL	5.000	116	46	70	5.400	0.080	0.080	...	7.200	-0.020	-0.100	0.000	19.200	0.410	57.000
2	Andrew	Abbott	CIN	41.200	674	240	434	2.380	0.290	0.080	...	4.370	1.200	0.900	1.800	17.900	0.430	87.000
3	Cory	Abbott	WSN	17.000	294	112	182	4.240	0.180	0.120	...	4.850	-0.250	0.000	0.100	18.100	0.350	97.000
4	Bryan	Abreu	HOU	42.000	746	282	464	2.790	0.370	0.100	...	2.890	-0.090	0.600	0.800	19.900	0.340	139.000

```
col_names = ['FirstName', 'LastName', 'Team', 'IP', 'Pitches','Balls', 'Strikes', 'ERA', 'K%', 'BB%', 'HR/9', 'FIP', 'ERA-', 'xFIP', 'WPA' , 'WAR', 'RA9-WAR', 'Pace (pi)', 'HardHit%', 'St


df.columns = col_names


# Look at the first 5 rows of data
df.head()
```

	FirstName	LastName	Team	IP	Pitches	Balls	Strikes	ERA	K%	BB%	...	xFIP	WPA	WAR	RA9-WAR	Pace (pi)
0	Jose	A. Ferrer	WSN	4.100	63	29	34	0.000	0.310	0.190	...	3.430	0.070	0.100	0.300	18.900
1	Fernando	Abad	COL	5.000	116	46	70	5.400	0.080	0.080	...	7.200	-0.020	-0.100	0.000	19.200
2	Andrew	Abbott	CIN	41.200	674	240	434	2.380	0.290	0.080	...	4.370	1.200	0.900	1.800	17.900
3	Cory	Abbott	WSN	17.000	294	112	182	4.240	0.180	0.120	...	4.850	-0.250	0.000	0.100	18.100
4	Bryan	Abreu	HOU	42.000	746	282	464	2.790	0.370	0.100	...	2.890	-0.090	0.600	0.800	19.900

```
df.isnull().sum()
```

```
FirstName          0
LastName           0
Team               0
IP                 0
Pitches            0
Balls              0
Strikes            0
ERA                0
K%                 0
BB%                0
HR/9               0
FIP                0
ERA-               0
xFIP               0
WPA                0
WAR                0
RA9-WAR            0
Pace (pi)          0
HardHit%           0
Stuff+            17
playerid           0
mlbamid            0
TimerViolations    0
dtype: int64
```

```
print(df.shape)
```

```
(751, 23)
```

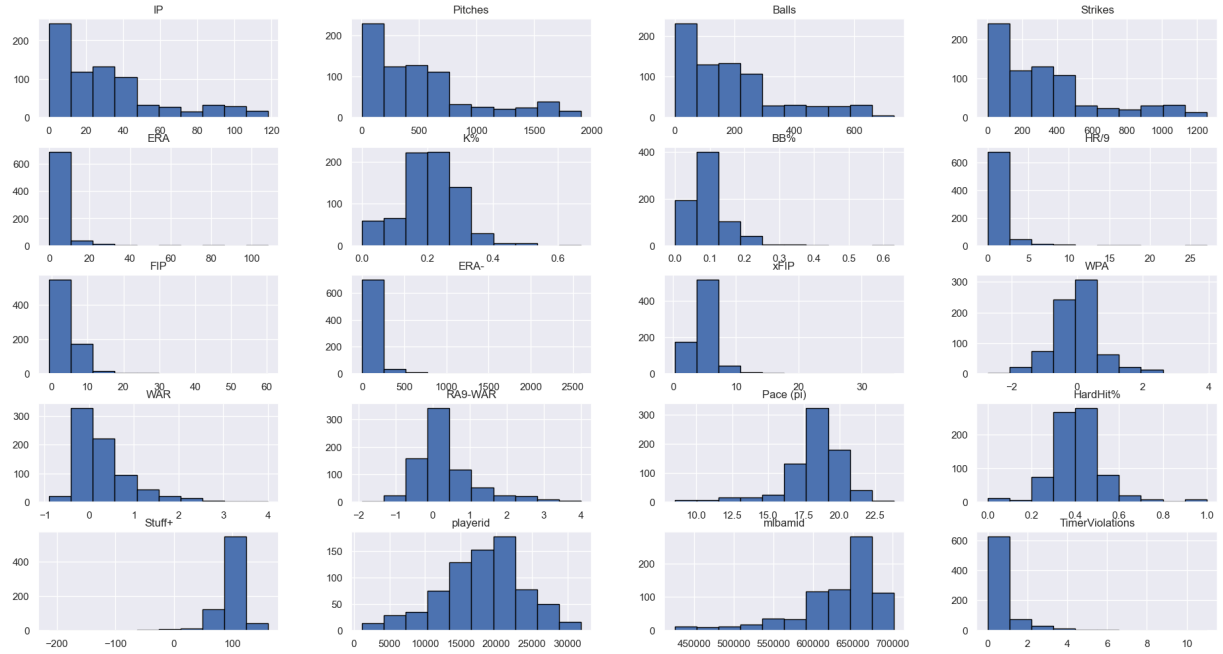
```
print(df.dtypes)
```

```
FirstName          object
LastName           object
Team               object
IP                float64
Pitches             int64
Balls               int64
Strikes             int64
ERA               float64
K%                float64
BB%               float64
HR/9              float64
FIP               float64
ERA-                int64
xFIP              float64
WPA               float64
WAR               float64
RA9-WAR           float64
Pace (pi)         float64
HardHit%          float64
Stuff+            float64
playerid            int64
mlbamid             int64
TimerViolations     int64
dtype: object
```

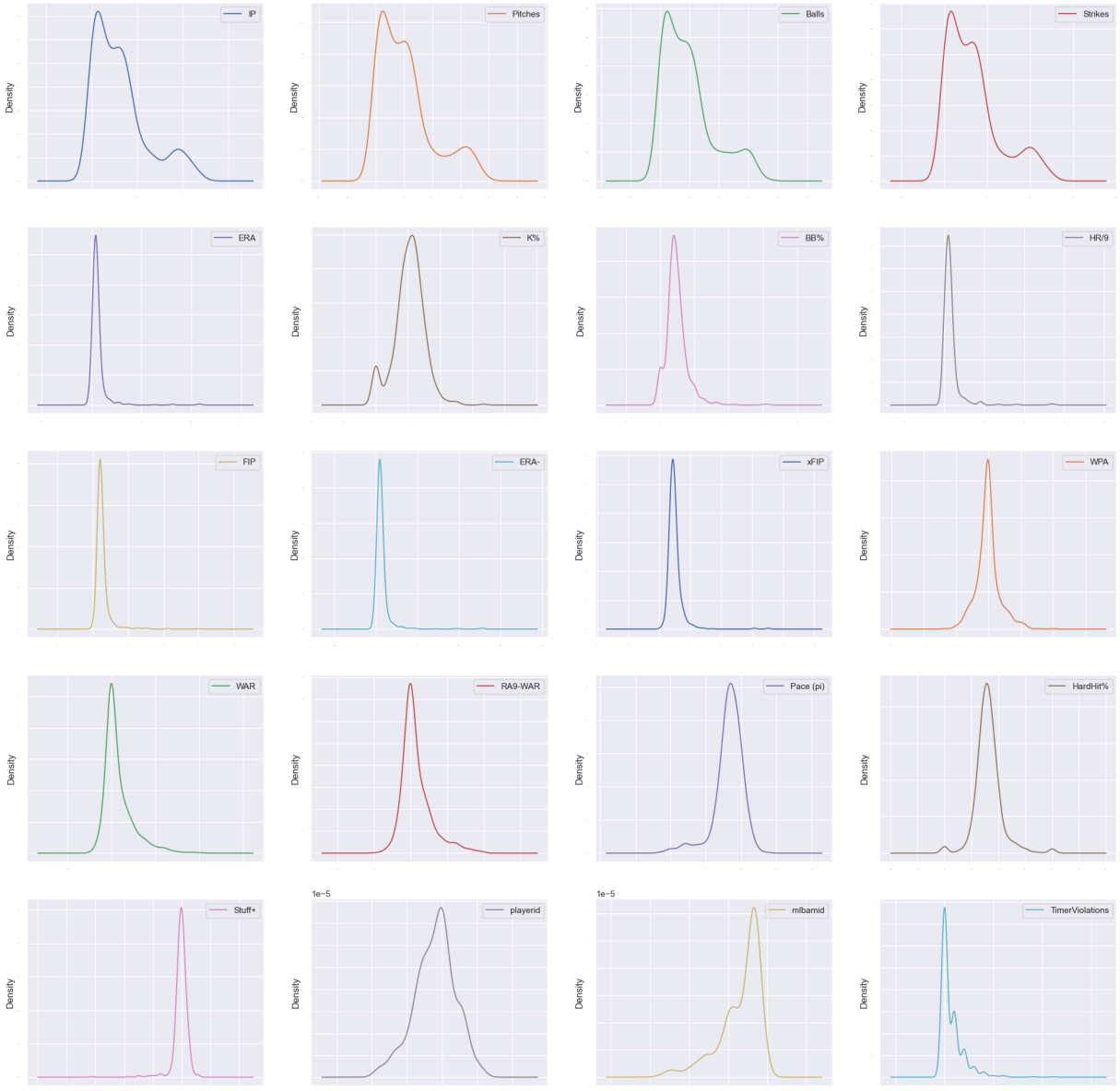
```
print(df.describe())
```

```
                IP      Pitches       Balls      Strikes         ERA  \
count  751.000000   751.000000  751.000000   751.000000  751.000000
mean    31.479893   528.605859  191.234354   337.371505    5.997909
std     29.393286   478.460422  171.259673   308.717164    9.177736
min      0.100000     4.000000    0.000000     3.000000    0.000000
25%      7.150000   137.500000   52.000000    83.500000    3.035000
50%     25.000000   423.000000  154.000000   273.000000    4.310000
75%     42.100000   697.000000  253.000000   445.500000    6.170000
max    118.100000  1906.000000  733.000000  1255.000000  108.000000

                K%         BB%        HR/9         FIP         ERA-  \
count  751.000000  751.000000  751.000000  751.000000   751.000000
mean     0.207044    0.095792    1.484927    5.242836   140.025300
std      0.092906    0.063023    2.389082    4.389534   216.792443
min      0.000000    0.000000    0.000000   -0.710000     0.000000
25%      0.160000    0.060000    0.520000    3.410000    70.000000
50%      0.210000    0.090000    1.060000    4.360000   101.000000
75%      0.260000    0.120000    1.660000    5.515000   141.000000
max      0.670000    0.630000   27.000000   60.290000  2593.000000

              xFIP         WPA         WAR     RA9-WAR   Pace (pi)    HardHit%  \
count  751.000000  751.000000  751.000000  751.000000  751.000000  751.000000
mean     4.974434   -0.034394    0.316112    0.315712   18.245273    0.406884
std      2.685985    0.725944    0.659712    0.830096    2.073123    0.122728
min      0.290000   -2.720000   -0.900000   -1.900000    8.500000    0.000000
25%      3.825000   -0.370000   -0.100000   -0.100000   17.500000    0.350000
50%      4.540000   -0.020000    0.100000    0.100000   18.500000    0.400000
75%      5.355000    0.175000    0.500000    0.600000   19.500000    0.460000
max     35.080000    3.940000    4.000000    4.000000   23.800000    1.000000

           Stuff+      playerid        mlbamid  TimerViolations
count  734.000000    751.000000     751.000000       751.000000
mean    96.694823  17989.014647  630512.246338         0.699068
std     25.373705   5778.541275   53869.292443         1.168469
min   -213.000000   1157.000000  425794.000000         0.000000
25%     90.000000  14462.500000  605473.000000         0.000000
50%     99.000000  18454.000000  656266.000000         0.000000
75%    108.000000  21508.500000  669003.000000         1.000000
max    161.000000  31839.000000  701643.000000        11.000000
```

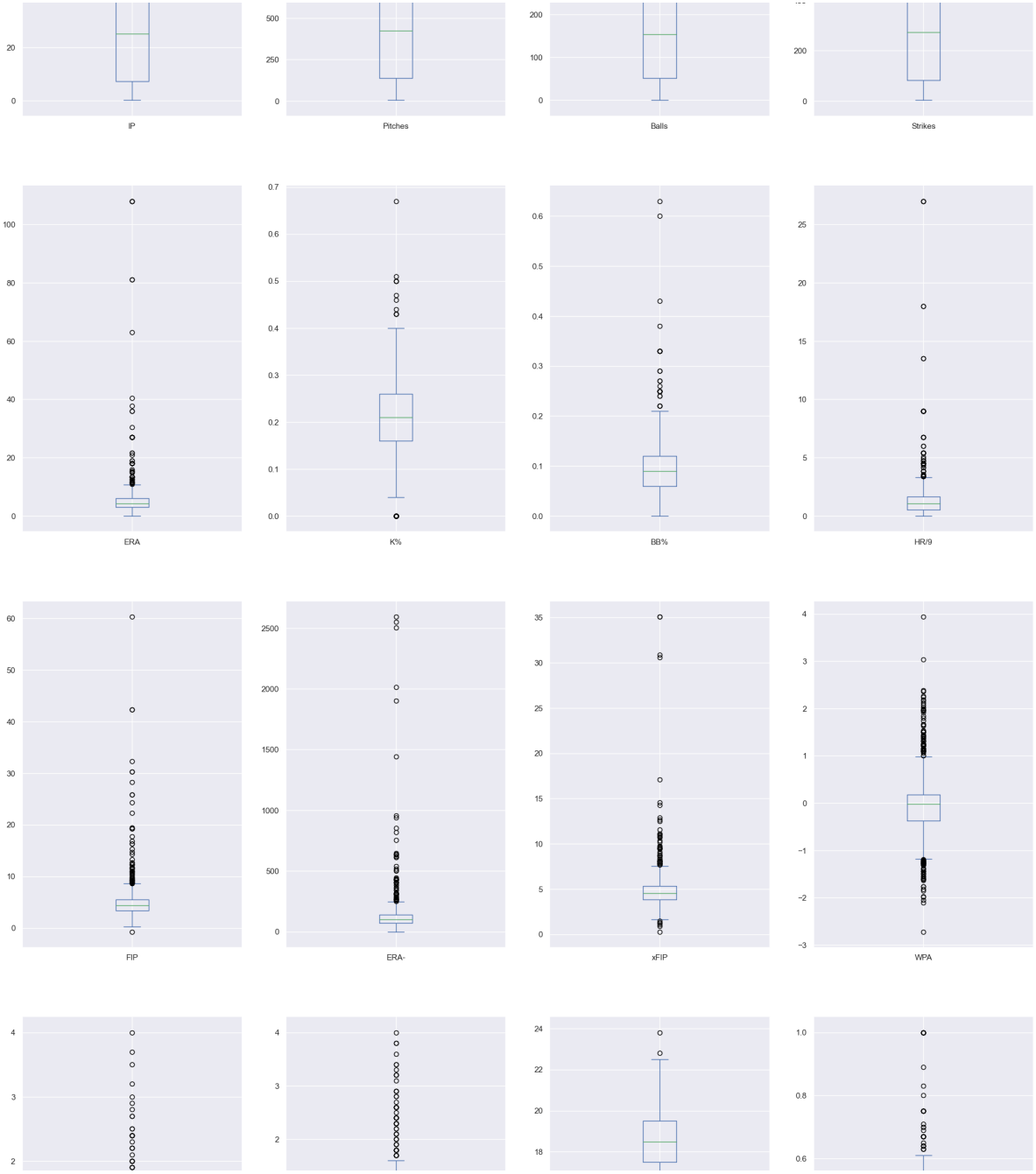
```
df.hist(edgecolor= 'black',figsize=(23,12))
plt.show()
```




```
df.plot(kind='density', subplots=True, layout= (5,4), sharex=False,legend=True, fontsize=1, figsize= (25,25))
plt.show()
```


IP
Pitches
Balls
Strikes
ERA
K%
BB%
HR/9
FIP
ERA-
xFIP
WPA
WAR
RA9-WAR
Pace (pi)
HardHit%
Stuff+
playerid
mlbamid
TimerViolations
Density
1e−5

```
df.plot(kind="box", subplots=True, layout=(5,4), sharex=False, figsize=(25,50))
plt.show()
```


IP
Pitches
Balls
Strikes
ERA
K%
BB%
HR/9
FIP
ERA-
xFIP
WPA

```
pd.options.display.float_format = '{:,.3f}'.format
```

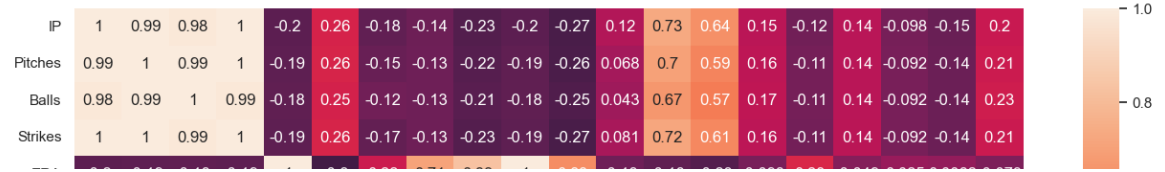
```
df.corr()
```

```
C:\Users\LordG\AppData\Local\Temp\ipykernel_6832\1134722465.py:1: FutureWarning: The default value of numeric_only in D
  df.corr()
```

	IP	Pitches	Balls	Strikes	ERA	K%	BB%	HR/9	FIP	ERA-	xFIP	WPA	WAR	RA9-WAR	
IP	1.000	0.995	0.982	0.996	-0.200	0.256	-0.181	-0.139	-0.232	-0.198	-0.272	0.117	0.730	0.640	(
Pitches	0.995	1.000	0.994	0.998	-0.189	0.261	-0.153	-0.132	-0.222	-0.187	-0.262	0.068	0.701	0.594	(
Balls	0.982	0.994	1.000	0.986	-0.183	0.253	-0.119	-0.130	-0.212	-0.182	-0.248	0.043	0.666	0.565	(
Strikes	0.996	0.998	0.986	1.000	-0.191	0.264	-0.172	-0.132	-0.227	-0.189	-0.269	0.081	0.717	0.607	(

```
plt.figure(figsize =(16,10))
sns.heatmap(df.corr(), annot=True)
plt.show()
```

```
C:\Users\LordG\AppData\Local\Temp\ipykernel_6832\1236886484.py:2: FutureWarning: The default value of numeric_only in D
  sns.heatmap(df.corr(), annot=True)
```



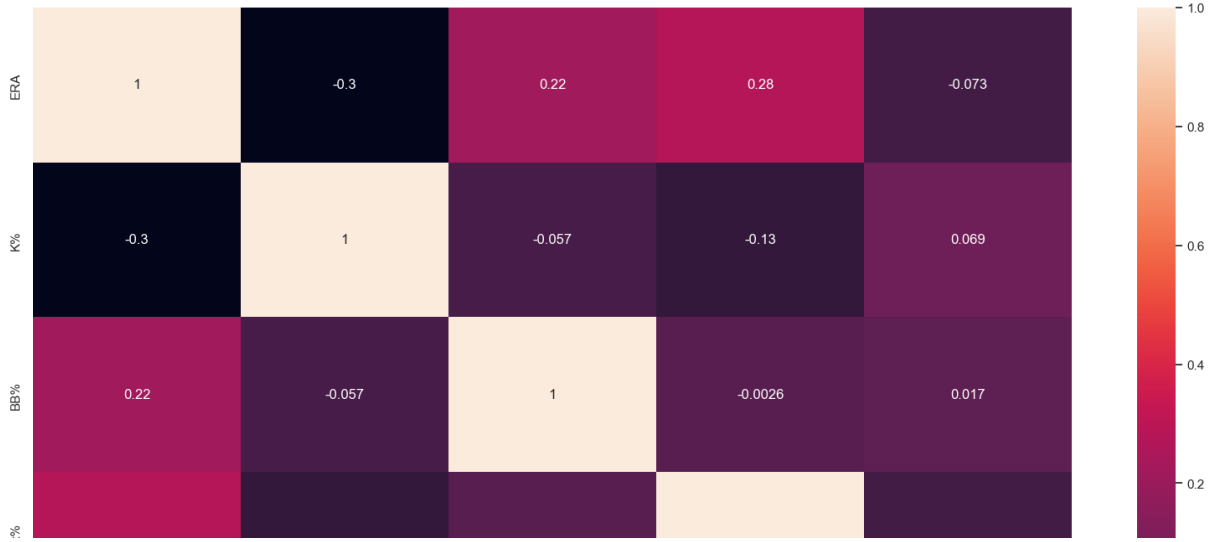

```
df2= df[['ERA','K%', 'BB%', 'HardHit%','TimerViolations']]
```

```
df2.corr()
```

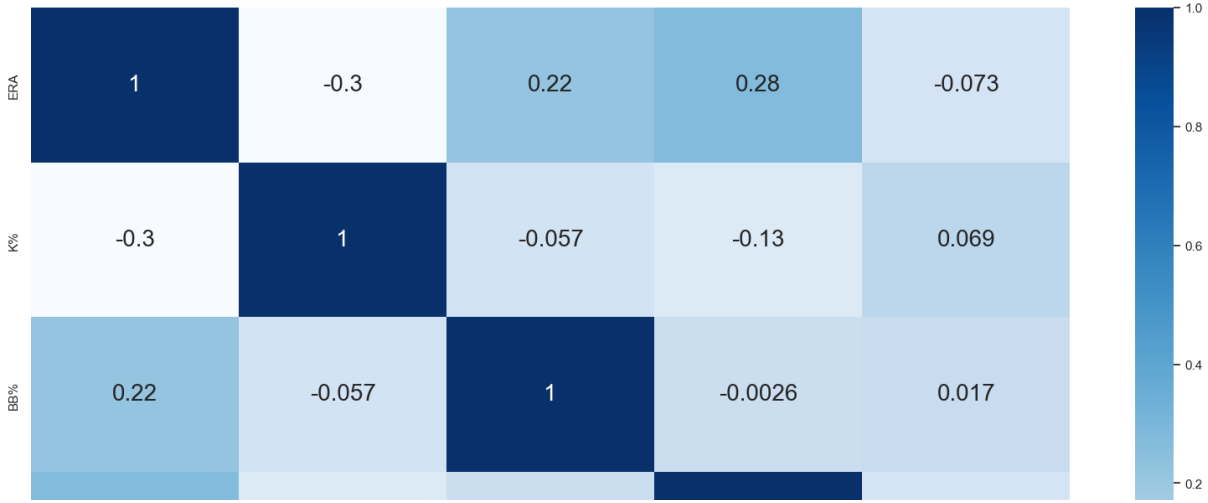
	ERA	K%	BB%	HardHit%	TimerViolations
ERA	1.000	-0.302	0.218	0.277	-0.073
K%	-0.302	1.000	-0.057	-0.127	0.069
BB%	0.218	-0.057	1.000	-0.003	0.017
HardHit%	0.277	-0.127	-0.003	1.000	-0.070
TimerViolations	-0.073	0.069	0.017	-0.070	1.000


```
sns.pairplot(df2, height=2);
plt.show()
```


```
plt.figure(figsize =(20,12))
sns.heatmap(df2.corr(), annot=True)
plt.show()
```

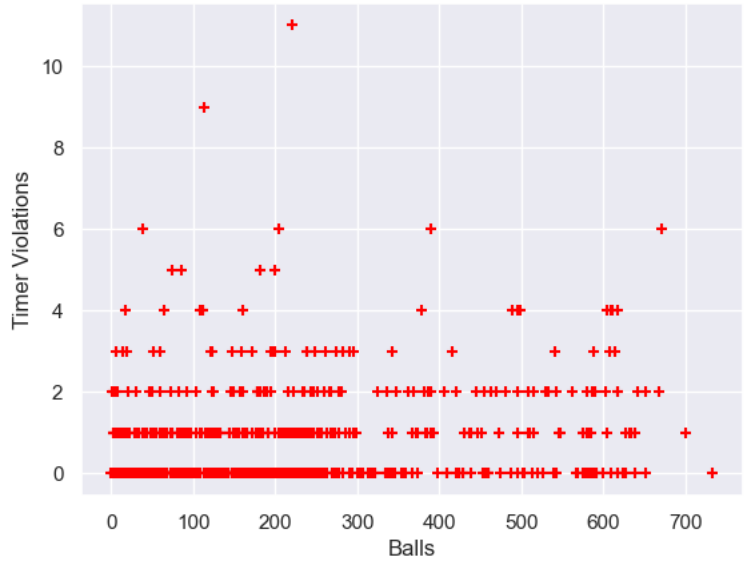



```
plt.figure(figsize =(20,12))
sns.heatmap(df2.corr(), cmap="Blues", annot=True, annot_kws={"fontsize":20})
plt.show()
```




```
%matplotlib inline
plt.xlabel('Balls')
plt.ylabel('Timer Violations')
plt.scatter(df.Balls, df.TimerViolations, color='red', marker='+')
```

```
<matplotlib.collections.PathCollection at 0x145b4e4b820>
```




```
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(df[['Balls']],df.TimerViolations)
```

```
reg.intercept_
```

```
0.40523677133269564
```

```
MLB2 = 'MLBtestData.csv'
```

```
df25= pd.read_csv (MLB2, header=None)
```

```
df25.head()
```

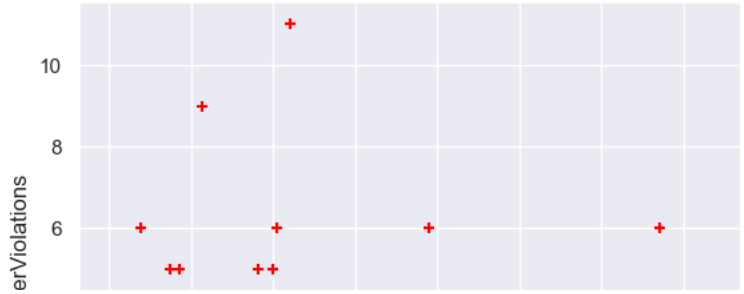
	0	1
0	29	1
1	46	0
2	240	0
3	112	4
4	282	3

```
col_names = ['Balls', 'TimerViolations']
df25.columns = col_names
df25.head()
```

	Balls	TimerViolations
0	29	1
1	46	0
2	240	0
3	112	4
4	282	3

```
%matplotlib inline
plt.xlabel('Balls')
plt.ylabel('TimerViolations')
plt.scatter(df25.Balls, df25.TimerViolations, color = 'red', marker = '+')
plt.plot(df25.Balls, reg.predict(df25[['Balls']]), color='blue')
```

```
[<matplotlib.lines.Line2D at 0x145b5a13340>]
```




```
# Store the dataframe values into a NumPy array
array= df25.values
X = array [:, 0:1]
Y = array [:,1]
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test= train_test_split(X,Y, test_size=0.2,random_state=seed)
model=LinearRegression()
model.fit(X_train, Y_train)
print ("Intercept:", model.intercept_)
print ("Coefficients:", model.coef_)
```

```
Intercept: 0.4601142799304338
Coefficients: [0.00128146]
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1)
```

```
LinearRegression
LinearRegression(n_jobs=1)
```

```
R_squared = model.score(X_test, Y_test)
print("R-squared: ", R_squared)
```

```
R-squared:  0.12462071203556313
```

```
model.predict([[13433]])
```

```
array([17.67391747])
```

```
num_folds = 10
seed = 7
kfold= KFold(n_splits=num_folds, random_state=seed, shuffle=True)
scoring = 'neg_mean_squared_error'
```

```
results = cross_val_score(model, X, Y, cv=kfold, scoring=scoring)
print("Average of all results from the K-fold Cross-Validation, using negative mean squared error:",results.mean())
```

```
Average of all results from the K-fold Cross-Validation, using negative mean squared error: -1.3061658132378515
```

```
MLBLinear = 'MLBLinearData.csv'
```

```
dfLinear= pd.read_csv (MLBLinear, header=None)
```

```
dfLinear.head()
```

	0	1	2	3	4	5	6
0	29	0.000	0.310	0.190	18.900	0.500	1
1	46	5.400	0.080	0.080	19.200	0.410	0
2	240	4.240	0.180	0.120	18.100	0.350	0
3	112	2.380	0.290	0.080	17.900	0.430	4
4	282	2.790	0.370	0.100	19.900	0.340	3

```
col_names = ['Balls', 'ERA', 'K%', 'BB%', 'Pace(pi)', 'HardHit%','TimerViolations']
dfLinear.columns = col_names
dfLinear.head()
```

	Balls	ERA	K%	BB%	Pace(pi)	HardHit%	TimerViolations
0	29	0.000	0.310	0.190	18.900	0.500	1
1	46	5.400	0.080	0.080	19.200	0.410	0
2	240	4.240	0.180	0.120	18.100	0.350	0
3	112	2.380	0.290	0.080	17.900	0.430	4
4	282	2.790	0.370	0.100	19.900	0.340	3

```
print(col_names.index("TimerViolations"))
```

```
6
```

```
# Store the dataframe values into a NumPy array
array= dfLinear.values
X = array [:, 0:6]
# For X (input) [:,6] --> All the rows and columns from 0 up to 6
Y = array [:,6]
# For Y (output) [:6] --> All the rows in the last column (TimerViolations)
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test= train_test_split(X,Y, test_size=0.2,random_state=seed)
model=LinearRegression()
model.fit(X_train, Y_train)
print ("Intercept:", model.intercept_)
print ("Coefficients:", model.coef_)
```

```
Intercept: 1.0187619505243988
Coefficients: [ 0.00119226 -0.00742564  0.81410498  0.83670486 -0.03188064 -0.41001797]
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1)
```

```
LinearRegression
LinearRegression(n_jobs=1)
```

```
R_squared = model.score(X_test, Y_test)
print("R-squared: ", R_squared)
```

```
R-squared:  0.11959744298850916
```

```
model.predict([[572,4.24,.27,.14,15,.24]])
```

```
array([1.42958228])
```

```
MLBFinal ='MLBAllData.csv'
```

```
dfFinal= pd.read_csv (MLBFinal, header=None)
```

```
dfFinal.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	4.100	63.000	29.000	34.000	0.000	0.310	0.190	0.000	3.060	0.000	3.430	0.070	0.100	0.300	18.900	0.500	7
1	5.000	116.000	46.000	70.000	5.400	0.080	0.080	3.600	8.890	108.000	7.200	-0.020	-0.100	0.000	19.200	0.410	5
2	17.000	294.000	112.000	182.000	4.240	0.180	0.120	1.060	4.880	96.000	4.850	-0.250	0.000	0.100	18.100	0.350	9
3	41.200	674.000	240.000	434.000	2.380	0.290	0.080	1.300	3.860	51.000	4.370	1.200	0.900	1.800	17.900	0.430	8
4	42.000	746.000	282.000	464.000	2.790	0.370	0.100	1.070	3.220	66.000	2.890	-0.090	0.600	0.800	19.900	0.340	13

```
col_names = ['IP', 'Pitches', 'Balls', 'Strikes', 'ERA', 'BB', 'K%','HR/9','FIP', 'ERA-', 'xFIP', 'WPA', 'WAR', 'RA9-WAR', 'Pace(pi)', 'HardHit%', 'Stuff+', 'TimerViolations']
dfFinal.columns = col_names
dfFinal.head()
```

	IP	Pitches	Balls	Strikes	ERA	BB	K%	HR/9	FIP	ERA-	xFIP	WPA	WAR	RA9-WAR	Pace(pi)	HardH
0	4.100	63.000	29.000	34.000	0.000	0.310	0.190	0.000	3.060	0.000	3.430	0.070	0.100	0.300	18.900	0.
1	5.000	116.000	46.000	70.000	5.400	0.080	0.080	3.600	8.890	108.000	7.200	-0.020	-0.100	0.000	19.200	0.
2	17.000	294.000	112.000	182.000	4.240	0.180	0.120	1.060	4.880	96.000	4.850	-0.250	0.000	0.100	18.100	0.
3	41.200	674.000	240.000	434.000	2.380	0.290	0.080	1.300	3.860	51.000	4.370	1.200	0.900	1.800	17.900	0.

```
print(col_names.index("TimerViolations"))
```

```
17
```

```
# Store the dataframe values into a NumPy array
array= dfFinal.values
X = array [:, 0:17]
# For X (input) [:,17] --> All the rows and columns from 0 up to 6
Y = array [:,17]
# For Y (output) [:17] --> All the rows in the last column (TimerViolations)
test_size = 0.33
seed = 25
X_train, X_test, Y_train, Y_test= train_test_split(X,Y, test_size=0.2,random_state=seed)
model=LinearRegression()
model.fit(X_train, Y_train)
print ("Intercept:", model.intercept_)
print ("Coefficients:", model.coef_)
```

```
Intercept: 1.1375789339208653
Coefficients: [-0.03580388  0.00238303  0.00446539 -0.00208235 -0.12389861 -0.21695569
  0.36903757  0.04195636 -0.04591418  0.00534106  0.03026241  0.06647235
  0.15826893 -0.01293539 -0.03453072 -0.42000623  0.00085386]
```

SINCE LINEAR REGRESSION CANT HANDLE MISSING VALUES, FOR THE 17 MISSING RECORDS OF STUFF+ REPRESENTING WHEN POSITION PLAYERS PITCHED, WE ASSIGNED THEM A VALUE OF 100 WHICH IS LEAGUE AVERAGE

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1)
```

```
LinearRegression
LinearRegression(n_jobs=1)
```

```
R_squared = model.score(X_test, Y_test)
print("R-squared: ", R_squared)
```

```
R-squared:  0.0007533132058813452
```

```
model.predict([[87.1,957,333,624,4.83,.25,.1, 1.4, 5,109, 4.7, 0.1,0.7, 0, 20, .33, 100]])
```

```
array([-0.201375])
```

```
MLBPitch = "MLBP&V.csv"
```

```
dfPitch= pd.read_csv (MLBPitch, header=None)
```

```
dfPitch.head()
```

	0	1
0	63	1
1	116	0
2	294	4
3	674	0
4	746	3