In [139…
```python
#Import all relevant libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import LinearRegression
from math import sqrt
from statsmodels.formula.api import ols
```

In [140…
```python
#See datatypes for each column
df = pd.read_excel(r"C:\Users\djbro\OneDrive\Desktop\NBA\sportsref_download (10).xls.xls
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 38 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Rk       30 non-null     int64  
 1   Season   30 non-null     object 
 2   Team     30 non-null     object 
 3   W        30 non-null     int64  
 4   Pace     30 non-null     float64
 5   ORtg     30 non-null     float64
 6   DRtg     30 non-null     float64
 7   MOV      30 non-null     float64
 8   SOS      30 non-null     float64
 9   W.1      30 non-null     int64  
 10  G        30 non-null     int64  
 11  W.2      30 non-null     int64  
 12  L        30 non-null     int64  
 13  W/L%     30 non-null     float64
 14  MP       30 non-null     int64  
 15  FG       30 non-null     int64  
 16  FGA      30 non-null     int64  
 17  2P       30 non-null     int64  
 18  2PA      30 non-null     int64  
 19  3P       30 non-null     int64  
 20  3PA      30 non-null     int64  
 21  FT       30 non-null     int64  
 22  FTA      30 non-null     int64  
 23  ORB      30 non-null     int64  
 24  DRB      30 non-null     int64  
 25  TRB      30 non-null     int64  
 26  AST      30 non-null     int64  
 27  STL      30 non-null     int64  
 28  BLK      30 non-null     int64  
 29  TOV      30 non-null     int64  
 30  PF       30 non-null     int64  
 31  PTS      30 non-null     int64  
 32  FG%      30 non-null     float64
 33  2P%      30 non-null     float64
 34  3P%      30 non-null     float64
 35  FT%      30 non-null     float64
 36  TS%      30 non-null     float64
 37  eFG%     30 non-null     float64
dtypes: float64(12), int64(24), object(2)
memory usage: 9.0+ KB
```

In [141…

```
#Return first 5 rows of the dataset
df.head()
```

Out[141]:

	Rk	Season	Team	W	Pace	ORtg	DRtg	MOV	SOS	W.1	...	BLK	TOV	PF	PTS	FG%	2P%	3P%
0	1	2022-23	BOS	21	98.9	119.3	112.4	6.89	0.22	21	...	142	383	564	3341	0.489	0.579	0.391
1	2	2022-23	MIL	19	99.0	112.2	107.6	4.58	-1.03	19	...	157	385	489	2913	0.456	0.532	0.350
2	3	2022-23	NOP	18	99.7	116.2	109.2	7.00	-0.11	18	...	116	383	513	3058	0.488	0.550	0.368
3	4	2022-23	MEM	18	100.5	114.0	110.3	3.81	-0.31	18	...	157	404	530	3119	0.468	0.530	0.360
4	5	2022-23	CLE	17	95.7	113.7	107.8	5.79	-0.54	17	...	119	418	535	3104	0.475	0.538	0.368

5 rows × 38 columns

In [142…

```
# There are not any missing values so we can move into univariate analysis
df.describe()
```

Out[142]:

	Rk	W	Pace	ORtg	DRtg	MOV	SOS	W.1	G	W
count	30.000000	30.00000	30.000000	30.000000	30.000000	30.000000	30.000000	30.00000	30.000000	30.000
mean	15.500000	13.60000	99.396667	112.823333	112.776667	0.028000	-0.003000	13.60000	27.200000	13.600
std	8.803408	3.55838	1.892541	2.750885	2.490490	4.126132	0.609149	3.55838	1.063501	3.558
min	1.000000	7.00000	95.700000	107.800000	107.600000	-9.960000	-1.120000	7.00000	25.000000	7.000
25%	8.250000	11.00000	98.125000	111.150000	111.500000	-1.500000	-0.370000	11.00000	26.250000	11.000
50%	15.500000	14.00000	99.350000	112.450000	112.600000	0.610000	0.150000	14.00000	27.000000	14.000
75%	22.750000	16.00000	100.850000	114.475000	114.100000	2.207500	0.297500	16.00000	28.000000	16.000
max	30.000000	21.00000	102.700000	119.300000	118.500000	7.000000	1.510000	21.00000	29.000000	21.000

8 rows × 36 columns

In [143…

```
#Create a Daily Fantasy Points column. This is our target statistic
df['FP'] = df["PTS"]+0.5*df['3P']+1.25*df["TRB"]+1.5*df["AST"]+2*df["STL"]+2*df["BLK"]-d
df.sort_values(by=['FP'],ascending=False).head(5)
```

Out[143]:

	Rk	Season	Team	W	Pace	ORtg	DRtg	MOV	SOS	W.1	...	TOV	PF	PTS	FG%	2P%	3P%	FT%
9	10	2022-23	UTA	15	100.0	116.4	115.0	1.41	0.05	15	...	458	619	3401	0.477	0.560	0.372	0.778
0	1	2022-23	BOS	21	98.9	119.3	112.4	6.89	0.22	21	...	383	564	3341	0.489	0.579	0.391	0.833
5	6	2022-23	BRK	17	98.5	113.8	112.2	1.55	0.04	17	...	430	636	3262	0.499	0.576	0.370	0.794
11	12	2022-23	IND	14	102.0	112.4	113.4	-0.96	-1.12	14	...	446	611	3210	0.453	0.525	0.365	0.789
3	4	2022-23	MEM	18	100.5	114.0	110.3	3.81	-0.31	18	...	404	530	3119	0.468	0.530	0.360	0.702

5 rows × 39 columns

In [144…

```
#Calculate new columns. Since the teams have played different numbers of games I thought
#would give better results
df['PPG'] = df['PTS']/df['G']
df['RPG'] = df['TRB']/df['G']
df['3PPG'] = df['3P']/df['G']
df['ASTPG'] = df['AST']/df['G']
df['TRPG'] = df['TRB']/df['G']
df['STLPG'] = df['STL']/df['G']
df['FPPG'] = df['FP']/df['G']
```

In [145…

```
#Reassign df to colums I am interested in
df = df[['Pace','ORtg','DRtg','PPG','3PPG','ASTPG','TRPG','STLPG','FPPG']]
df.head()
```

Out[145]:

	Pace	ORtg	DRtg	PPG	3PPG	ASTPG	TRPG	STLPG	FPPG
0	98.9	119.3	112.4	119.321429	16.178571	26.750000	42.464286	6.392857	229.866071
1	99.0	112.2	107.6	112.038462	13.038462	25.076923	48.538462	6.461538	227.038462
2	99.7	116.2	109.2	117.615385	11.115385	27.038462	45.192308	8.884615	232.182692
3	100.5	114.0	110.3	115.518519	12.148148	25.185185	48.666667	7.481481	231.833333
4	95.7	113.7	107.8	110.857143	11.607143	23.357143	43.071429	6.464286	212.035714

In [146…

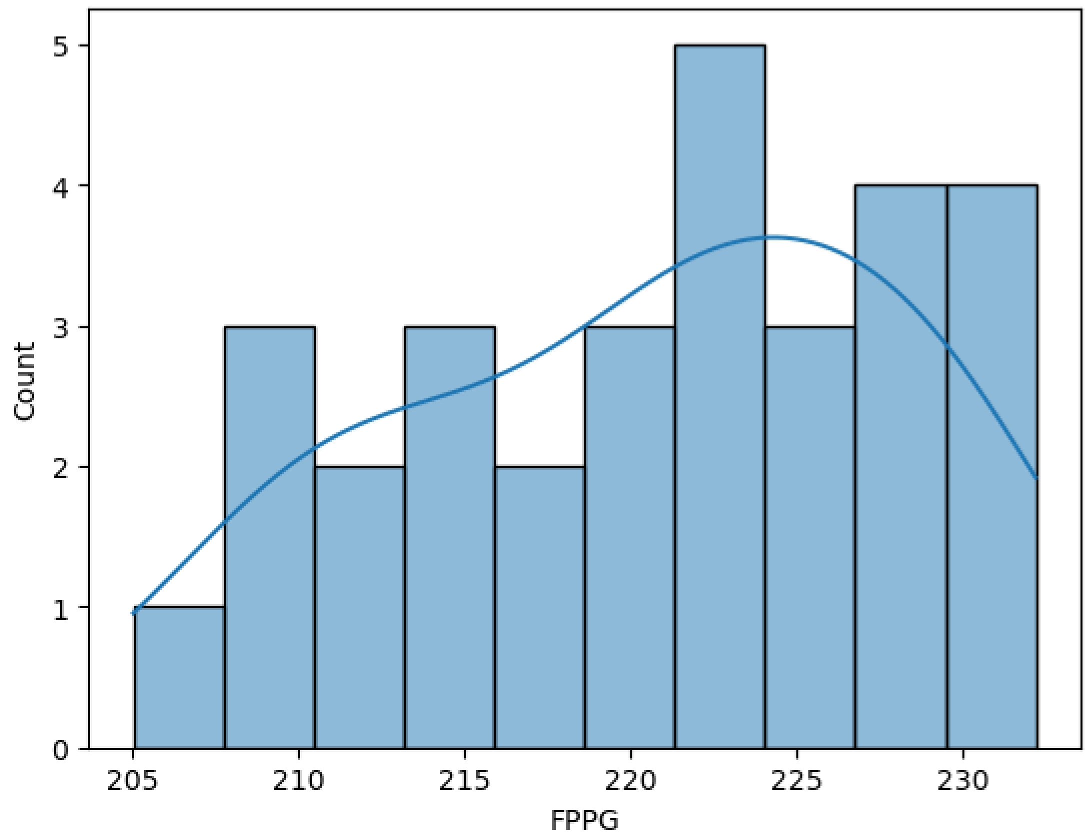
```
#Univariate Analysis
df.describe()
```

Out[146]:

	Pace	ORtg	DRtg	PPG	3PPG	ASTPG	TRPG	STLPG	FPPG
count	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000
mean	99.396667	112.823333	112.776667	113.070697	12.116539	24.914123	43.595836	7.376268	220.587798
std	1.892541	2.750885	2.490490	3.235884	1.735281	2.098745	2.197347	0.856062	7.643043
min	95.700000	107.800000	107.600000	107.857143	9.730769	21.357143	38.851852	6.178571	205.026786
25%	98.125000	111.150000	111.500000	110.504121	10.973214	23.468585	42.514881	6.867706	214.530716
50%	99.350000	112.450000	112.600000	112.812808	11.714191	24.662393	43.148148	7.135556	221.930396
75%	100.850000	114.475000	114.100000	115.462963	12.898148	26.717672	44.570437	7.775641	226.906161
max	102.700000	119.300000	118.500000	119.321429	16.296296	29.555556	48.666667	9.962963	232.182692

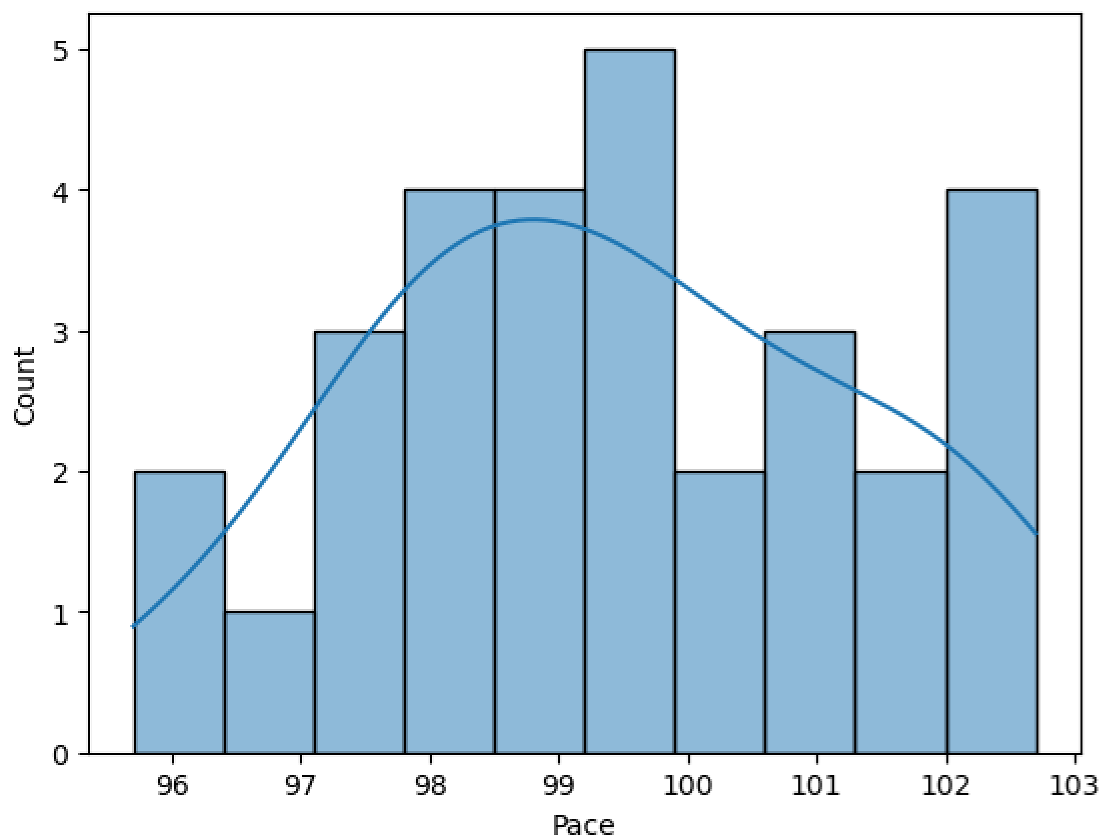
In [147…

```
#Univariate Analysis of Fantasy Points Per Game
sns.histplot(x=df['FPPG'], bins=10, kde=True);
```



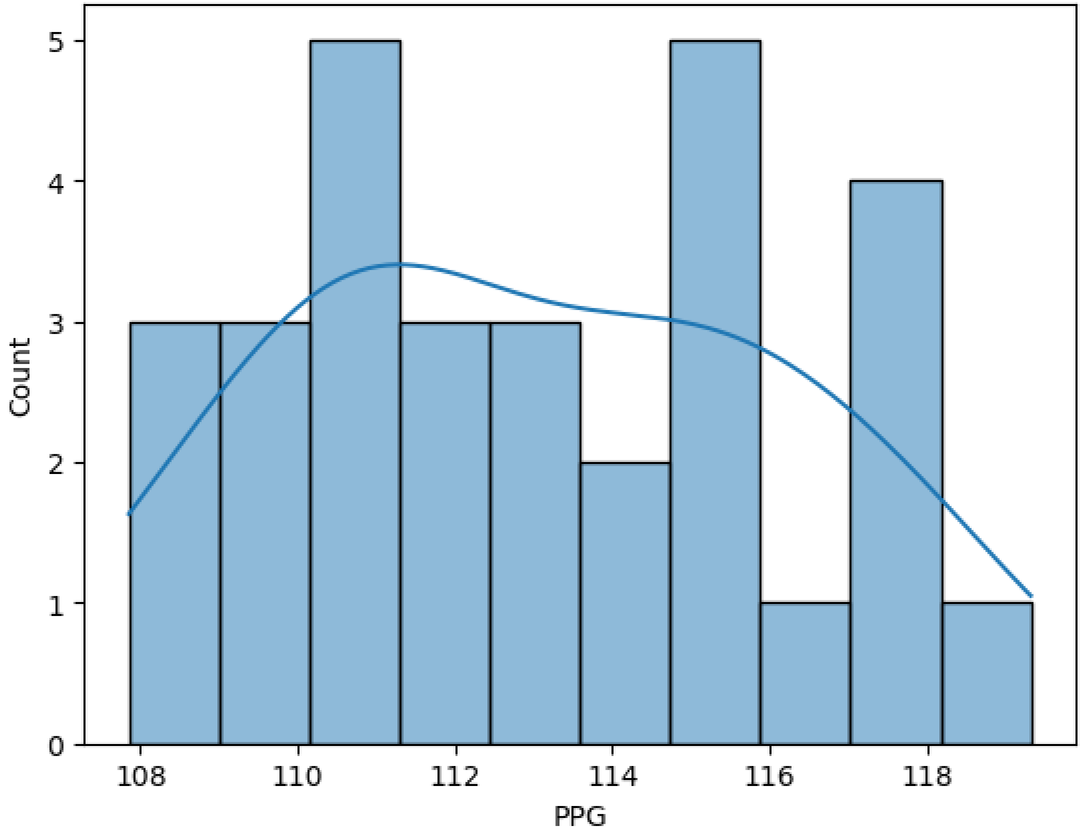
In [148…

```
#Univariate Analysis of Pace
sns.histplot(x=df['Pace'], bins=10, kde=True);
```



In [149…

```
#Univariate Analysis of Points Per Game
sns.histplot(x=df['PPG'], bins=10, kde=True);
```

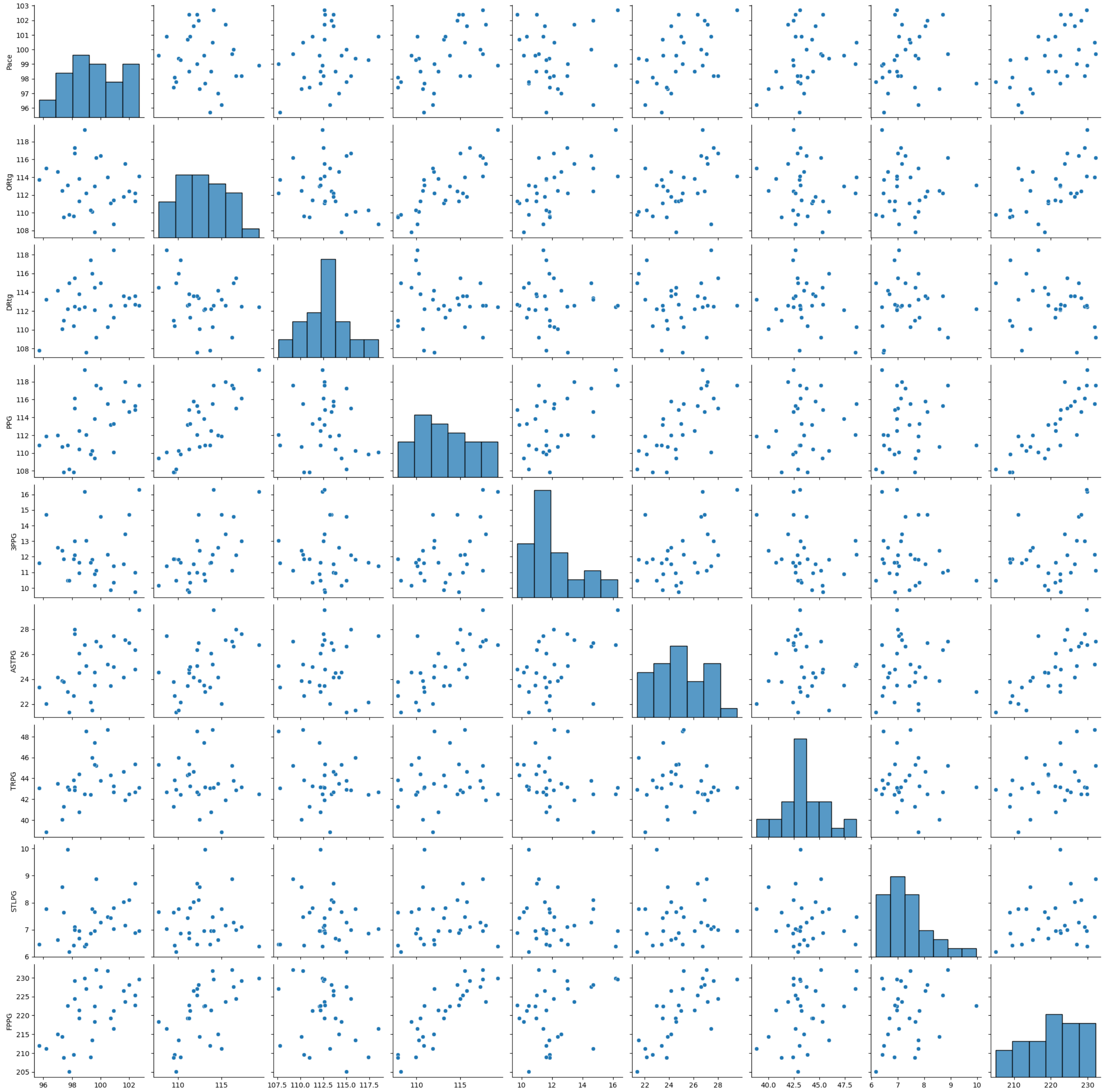


In [150…

```
# UNIVARIATE AND BIVARIATE Visualization via seaborn.
sns.pairplot(df)
```

Out[150]:

```
<seaborn.axisgrid.PairGrid at 0x2774b4d1460>
```



In [151…

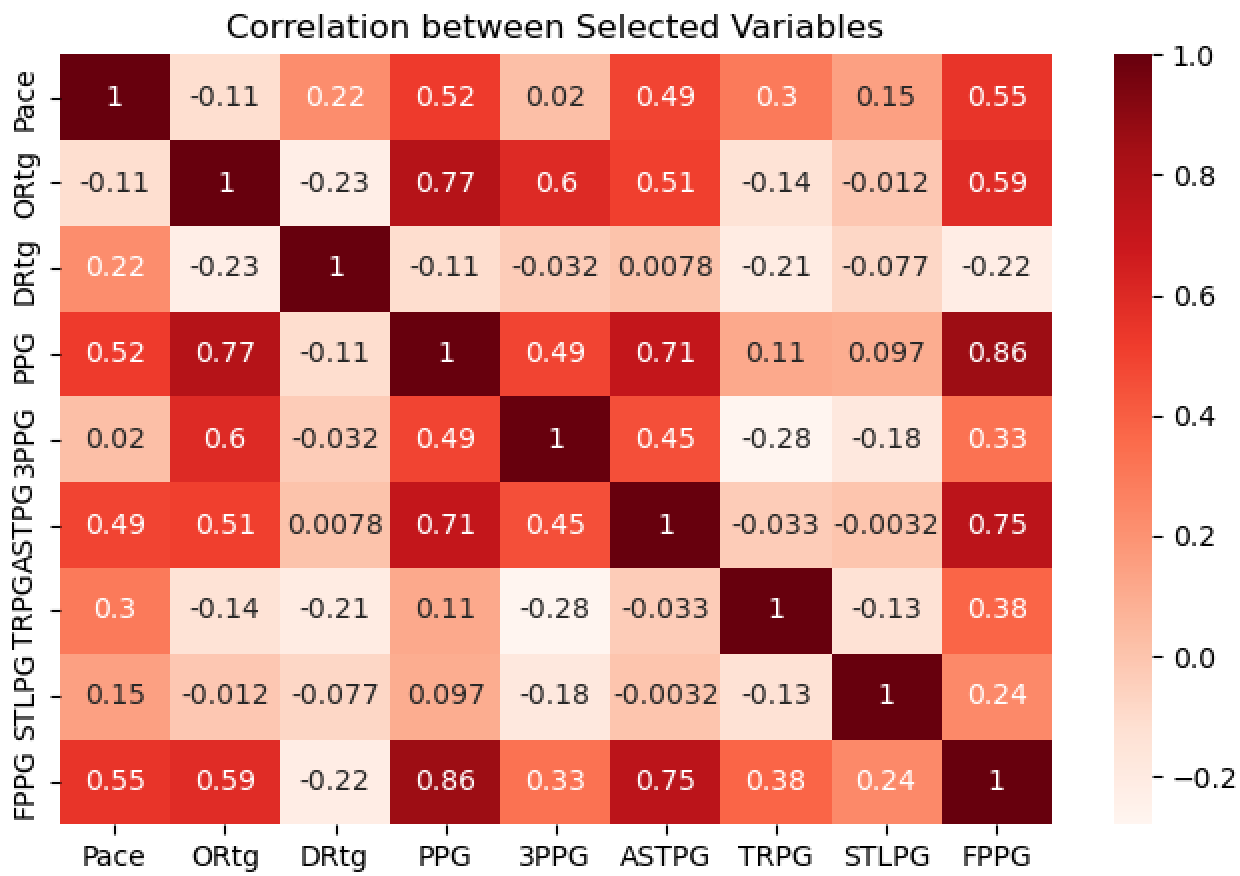
```
#Create a correlation matrix to show relationship between select variables
corr_matrix = df[['Pace','ORtg','DRtg','PPG','3PPG','ASTPG','TRPG','STLPG','FPPG']].corr
corr_matrix
```

Out[151]:

	Pace	ORtg	DRtg	PPG	3PPG	ASTPG	TRPG	STLPG	FPPG
Pace	1.000000	-0.114968	0.222096	0.523419	0.020336	0.488460	0.295302	0.148201	0.550522
ORtg	-0.114968	1.000000	-0.226664	0.770206	0.595495	0.507282	-0.138065	-0.011610	0.589064
DRtg	0.222096	-0.226664	1.000000	-0.107967	-0.031923	0.007836	-0.207162	-0.077423	-0.218013
PPG	0.523419	0.770206	-0.107967	1.000000	0.485146	0.710085	0.113923	0.096799	0.859269
3PPG	0.020336	0.595495	-0.031923	0.485146	1.000000	0.451282	-0.280458	-0.181416	0.325587
ASTPG	0.488460	0.507282	0.007836	0.710085	0.451282	1.000000	-0.032762	-0.003211	0.748917
TRPG	0.295302	-0.138065	-0.207162	0.113923	-0.280458	-0.032762	1.000000	-0.133479	0.378905
STLPG	0.148201	-0.011610	-0.077423	0.096799	-0.181416	-0.003211	-0.133479	1.000000	0.238537
FPPG	0.550522	0.589064	-0.218013	0.859269	0.325587	0.748917	0.378905	0.238537	1.000000

In [152…

```
#Create a heatmap to visualize correlation
plt.figure(figsize=[8,5])
sns.heatmap(corr_matrix,annot=True,cmap='Reds')
plt.title("Correlation between Selected Variables")
plt.show()
```



	Pace	ORtg	DRtg	PPG	3PPG	ASTPG	TRPG	STLPG	FPPG
Pace	1	-0.11	0.22	0.52	0.02	0.49	0.3	0.15	0.55
ORtg	-0.11	1	-0.23	0.77	0.6	0.51	-0.14	-0.012	0.59
DRtg	0.22	-0.23	1	-0.11	-0.032	0.0078	-0.21	-0.077	-0.22
PPG	0.52	0.77	-0.11	1	0.49	0.71	0.11	0.097	0.86
3PPG	0.02	0.6	-0.032	0.49	1	0.45	-0.28	-0.18	0.33
ASTPG	0.49	0.51	0.0078	0.71	0.45	1	-0.033	-0.0032	0.75
TRPG	0.3	-0.14	-0.21	0.11	-0.28	-0.033	1	-0.13	0.38
STLPG	0.15	-0.012	-0.077	0.097	-0.18	-0.0032	-0.13	1	0.24
FPPG	0.55	0.59	-0.22	0.86	0.33	0.75	0.38	0.24	1

In [153…

```
#Feature Selection.
#Start easy . We could look a Pearson correlation above. Also just look at variances
```

In [154…

```
#Look to see which features have low variance. We can likely drop STLPG but we will hold
df.var()
```

Out[154]:

```
Pace      3.581713
ORtg      7.567368
DRtg      6.202540
PPG      10.470944
3PPG      3.011200
ASTPG     4.404731
TRPG      4.828334
STLPG     0.732843
FPPG     58.416105
dtype: float64
```

In [155…

```
#Look at correlation of feature variables with target variable
abs(df.corr()["FPPG"])
```

Out[155]:

```
Pace     0.550522
ORtg     0.589064
DRtg     0.218013
PPG      0.859269
3PPG     0.325587
ASTPG    0.748917
TRPG     0.378905
STLPG    0.238537
```

```
FPPG     1.000000
Name: FPPG, dtype: float64
```

In [156…

```
#Create features and target dfs
X = df.drop(columns='FPPG')
y = df.FPPG
```

In [157…

```
#Using cross validation to ensure there is not overfitting, and then checking for RMSE a
#of features with Fantasy Points Per Game. The R squared of 0.67 means 67% of variabilit
#and PPG
cv = KFold(n_splits=10, random_state=0,shuffle=True)
classifier_pipeline = make_pipeline(StandardScaler(), KNeighborsRegressor(n_neighbors=10
vals = [0.1,0.2,0.3,0.4,0.5,0.6,0.7]
for val in vals:
    features = abs(df.corr()["FPPG"][abs(df.corr()["FPPG"])>val].drop('FPPG')).index.tol

    X = df.drop(columns='FPPG')
    X=X[features]

    print(features)

    y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
    print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),2)))
    print("R_squared: " + str(round(r2_score(y,y_pred),2)))
```

```
['Pace', 'ORtg', 'DRtg', 'PPG', '3PPG', 'ASTPG', 'TRPG', 'STLPG']
RMSE: 4.3
R_squared: 0.67
['Pace', 'ORtg', 'DRtg', 'PPG', '3PPG', 'ASTPG', 'TRPG', 'STLPG']
RMSE: 4.3
R_squared: 0.67
['Pace', 'ORtg', 'PPG', '3PPG', 'ASTPG', 'TRPG']
RMSE: 4.52
R_squared: 0.64
['Pace', 'ORtg', 'PPG', 'ASTPG']
RMSE: 4.51
R_squared: 0.64
['Pace', 'ORtg', 'PPG', 'ASTPG']
RMSE: 4.51
R_squared: 0.64
['PPG', 'ASTPG']
RMSE: 4.17
R_squared: 0.69
['PPG', 'ASTPG']
RMSE: 4.17
R_squared: 0.69
```

In [158…

```
#Linear Regression Model with FPPG and PPG and ASTPG
df = df[['PPG','ASTPG','FPPG']]
FFPG_vs_features = ols('FPPG ~ PPG + ASTPG',data=df)
FFPG_vs_features = FFPG_vs_features.fit()
print(FFPG_vs_features.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   FPPG   R-squared:                       0.777
Model:                            OLS   Adj. R-squared:                  0.761
Method:                 Least Squares   F-statistic:                     47.09
Date:                Tue, 13 Dec 2022   Prob (F-statistic):           1.58e-09
Time:                        21:34:22   Log-Likelihood:                -80.553
No. Observations:                  30   AIC:                             167.1
Df Residuals:                      27   BIC:                             171.3
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
```

```
------------------------------------------------------------------------------
Intercept     18.7879     27.421      0.685      0.499     -37.476      75.052
PPG            1.5601      0.305      5.120      0.000       0.935       2.185
ASTPG          1.0193      0.470      2.169      0.039       0.055       1.983
==============================================================================
Omnibus:                        1.916   Durbin-Watson:                   2.002
Prob(Omnibus):                  0.384   Jarque-Bera (JB):                1.456
Skew:                           0.534   Prob(JB):                        0.483
Kurtosis:                       2.852   Cond. No.                     4.65e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
[2] The condition number is large, 4.65e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [159…

```python
##Linear Regression Model with FPPG and PPG
FFPG_vs_features = ols('FPPG ~ PPG',data=df)
FFPG_vs_features = FFPG_vs_features.fit()
print(FFPG_vs_features.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   FPPG   R-squared:                       0.738
Model:                            OLS   Adj. R-squared:                  0.729
Method:                 Least Squares   F-statistic:                     79.01
Date:                Tue, 13 Dec 2022   Prob (F-statistic):           1.21e-09
Time:                        21:34:22   Log-Likelihood:                -82.963
No. Observations:                  30   AIC:                             169.9
Df Residuals:                      28   BIC:                             172.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -8.8963     25.827     -0.344      0.733     -61.802      44.009
PPG            2.0296      0.228      8.889      0.000       1.562       2.497
==============================================================================
Omnibus:                        0.257   Durbin-Watson:                   2.116
Prob(Omnibus):                  0.879   Jarque-Bera (JB):                0.387
Skew:                           0.188   Prob(JB):                        0.824
Kurtosis:                       2.590   Cond. No.                     4.02e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
[2] The condition number is large, 4.02e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```