In [1]:
```
#Create a multiple Linear Regression that predicts Fantasy Points For DFS using only inf
#What information is available from the sportsbooks? How can we create a multiple linear
#this information and this information only
#Sportsbooks like DraftKings give me bets for player props such as O/U on
#Points, Assists, and Rebounds
```

In [2]:
```
#This code will create a multiple linear regression model, use RFE to
#select the most important features, fit the model to the training
#data, make predictions on the testing data, and evaluate the
#model's performance using metrics such as mean absolute error and
#mean squared error. You can customize and fine-tune this code as
#needed to meet your specific goals and requirements.
```

In [3]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
```

In [4]:
```
#Load in raw NBA Player Game by Game data
df = pd.read_excel(r'C:\Users\djbro\OneDrive\Desktop\DFS\NBA\Updated_Season_GameLogs\NBA
```

In [5]:
```
#Feature Engineering of dataset

#Create double double column for dataframe
df['DD'] = (df['PTS']>=10)&(df['AST']>=10)|(df['PTS']>=10)&(df['TRB']>=10)|(df['TRB']>=1

#creates triple double column for dataframe
df['TD'] = (df['PTS']>=10)&(df['AST']>=10)&(df['TRB']>=10)

#ensures a player can not get points for a triple double and double double in one game
df['DD'] = (df['DD']==True)&(df['TD']==False)

#Change data types of 'DD' and 'TD'
df['DD'] = df['DD'].astype(int)
df['TD'] = df['TD'].astype(int)
```

In [6]:
```
#Sort dataframe for easier viewing of upcoming calculations
df = df.sort_values(['Player', 'Date'], ascending= [True, True], ignore_index=True)
```

In [7]:
```
#calculates Draft Kings Fantasy Points totals for each game
df['FP'] = df["PTS"]+0.5*df['3P']+1.25*df["TRB"]+1.5*df["AST"]+2*df["STL"]+2*df["BLK"]+1
```

In [8]:
```
print(df.columns)
print(df.head(50))
```

```
Index(['Unnamed: 0', 'Rk', 'Player', 'PTS', 'PTS.1', 'Date', 'Age', 'Team',
       'Unnamed: 7', 'Opp', 'Result', 'GS', 'MP', 'FG', 'FGA', 'FG%', '2P',
       '2PA', '2P%', '3P', '3PA', '3P%', 'FT', 'FTA', 'FT%', 'TS%', 'ORB',
       'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS.2', 'GmSc', 'BPM',
       'Pos.', 'Unnamed: 6', 'DD', 'TD', 'FP'],
      dtype='object')
    Unnamed: 0    Rk        Player  PTS  PTS.1       Date     Age Team  \
0          170  6171    A.J. Green    0      0 2022-10-22  23-025  MIL
1          108  5109    A.J. Green    3      3 2022-11-16  23-050  MIL
2          171  6172    A.J. Green    0      0 2022-11-21  23-055  MIL
3          172  6173    A.J. Green    0      0 2022-11-25  23-059  MIL
4          173  6174    A.J. Green    0      0 2022-11-27  23-061  MIL
5           73    74    A.J. Green    8      8 2022-12-03  23-067  MIL
6           64    65    A.J. Green   12     12 2022-12-05  23-069  MIL
7           22   223    A.J. Green    0      0 2022-12-07  23-071  MIL
8           96    97    A.J. Green    2      2 2022-12-13  23-077  MIL
```

9	41	42	A.J. Green	10	10	2022-12-15	23-079	MIL
10	113	114	A.J. Green	3	3	2022-12-17	23-081	MIL
11	22	5623	A.J. Lawson	2	2	2022-11-16	22-124	MIN
12	65	3466	AJ Griffin	8	8	2022-10-23	19-059	ATL
13	54	2855	AJ Griffin	10	10	2022-10-28	19-064	ATL
14	180	6181	AJ Griffin	0	0	2022-10-29	19-065	ATL
15	114	5515	AJ Griffin	2	2	2022-10-31	19-067	ATL
16	113	5114	AJ Griffin	3	3	2022-11-02	19-069	ATL
17	182	583	AJ Griffin	24	24	2022-11-07	19-074	ATL
18	114	5115	AJ Griffin	3	3	2022-11-09	19-076	ATL
19	115	5116	AJ Griffin	3	3	2022-11-10	19-077	ATL
20	145	3146	AJ Griffin	9	9	2022-11-12	19-079	ATL
21	176	3777	AJ Griffin	7	7	2022-11-14	19-081	ATL
22	69	4070	AJ Griffin	6	6	2022-11-16	19-083	ATL
23	195	1396	AJ Griffin	17	17	2022-11-19	19-086	ATL
24	196	1397	AJ Griffin	17	17	2022-11-21	19-088	ATL
25	180	2381	AJ Griffin	12	12	2022-11-23	19-090	ATL
26	0	2601	AJ Griffin	11	11	2022-11-25	19-092	ATL
27	66	3467	AJ Griffin	8	8	2022-11-27	19-094	ATL
28	116	5117	AJ Griffin	3	3	2022-11-28	19-095	ATL
29	21	22	AJ Griffin	24	24	2022-12-02	19-099	ATL
30	72	73	AJ Griffin	11	11	2022-12-05	19-102	ATL
31	111	112	AJ Griffin	9	9	2022-12-07	19-104	ATL
32	95	96	AJ Griffin	10	10	2022-12-09	19-106	ATL
33	30	31	AJ Griffin	17	17	2022-12-11	19-108	ATL
34	46	47	AJ Griffin	13	13	2022-12-12	19-109	ATL
35	63	64	AJ Griffin	15	15	2022-12-14	19-111	ATL
36	69	70	AJ Griffin	13	13	2022-12-16	19-113	ATL
37	31	32	AJ Griffin	19	19	2022-12-19	19-116	ATL
38	62	63	AJ Griffin	14	14	2022-12-21	19-118	ATL
39	163	764	Aaron Gordon	22	22	2022-10-19	27-033	DEN
40	42	2843	Aaron Gordon	10	10	2022-10-21	27-035	DEN
41	190	2591	Aaron Gordon	11	11	2022-10-22	27-036	DEN
42	35	436	Aaron Gordon	26	26	2022-10-24	27-038	DEN
43	50	4051	Aaron Gordon	6	6	2022-10-26	27-040	DEN
44	104	5105	Aaron Gordon	3	3	2022-10-28	27-042	DEN
45	48	1249	Aaron Gordon	18	18	2022-10-30	27-044	DEN
46	178	379	Aaron Gordon	27	27	2022-11-03	27-048	DEN
47	55	3456	Aaron Gordon	8	8	2022-11-05	27-050	DEN
48	136	3137	Aaron Gordon	9	9	2022-11-07	27-052	DEN
49	49	1250	Aaron Gordon	18	18	2022-11-09	27-054	DEN

	Unnamed: 7	Opp	...	TOV	PF	PTS.2	GmSc	BPM	Pos.	Unnamed: 6	DD	TD \
0	NaN	HOU	...	0	0	0.0	0.7	10.3	G	NaN	0	0
1	NaN	CLE	...	0	2	3.0	2.1	-4.3	G	NaN	0	0
2	NaN	POR	...	0	0	0.0	0.0	-8.1	G	NaN	0	0
3	NaN	CLE	...	0	1	0.0	-0.1	-9.0	G	NaN	0	0
4	NaN	DAL	...	0	1	0.0	-1.1	-41.6	G	NaN	0	0
5	@	CHO	...	1	1	8.0	6.9	7.6	G	NaN	0	0
6	@	ORL	...	0	2	12.0	8.6	17.4	G	NaN	0	0
7	NaN	SAC	...	0	0	0.0	0.0	-9.2	G	NaN	0	0
8	NaN	GSW	...	0	1	2.0	1.3	-3.1	G	NaN	0	0
9	@	MEM	...	1	1	10.0	6.7	2.4	G	NaN	0	0
10	NaN	UTA	...	0	2	NaN	1.9	6.7	G	NaN	0	0
11	@	ORL	...	0	1	2.0	1.6	4.8	G	NaN	0	0
12	NaN	CHO	...	0	0	8.0	8.4	42.7	F	NaN	0	0
13	@	DET	...	0	2	10.0	6.6	16.8	F	NaN	0	0
14	@	MIL	...	0	0	0.0	0.0	-13.7	F	NaN	0	0
15	@	TOR	...	0	1	2.0	1.2	-8.6	F	NaN	0	0
16	@	NYK	...	0	0	3.0	3.4	5.1	F	NaN	0	0
17	NaN	MIL	...	1	2	24.0	21.0	11.9	F	NaN	0	0
18	NaN	UTA	...	0	1	3.0	0.8	-10.9	F	NaN	0	0
19	NaN	PHI	...	1	2	3.0	1.8	0.0	F	NaN	0	0
20	@	PHI	...	1	0	9.0	4.6	2.1	F	NaN	0	0
21	@	MIL	...	2	1	7.0	3.3	-3.8	F	NaN	0	0
22	NaN	BOS	...	2	1	6.0	2.3	-10.0	F	NaN	0	0

```
23        NaN  TOR  ...   0   2   17.0  14.0    4.0      F          NaN   0    0
24          @  CLE  ...   2   3   17.0  13.6    4.9      F          NaN   0    0
25        NaN  SAC  ...   1   0   12.0  10.7    5.3      F          NaN   0    0
26          @  HOU  ...   1   2   11.0   9.9    5.6      F          NaN   0    0
27        NaN  MIA  ...   0   1    8.0   1.3  -10.7      F          NaN   0    0
28          @  PHI  ...   1   2    3.0  -0.6  -13.8      F          NaN   0    0
29        NaN  DEN  ...   2   0   24.0  20.2    7.1      F          NaN   0    0
30        NaN  OKC  ...   1   2   11.0   6.3   -1.8      F          NaN   0    0
31          @  NYK  ...   1   2    9.0   2.0   -9.9      F          NaN   0    0
32          @  BRK  ...   3   0   10.0   3.9   -9.4      F          NaN   0    0
33        NaN  CHI  ...   1   2   17.0   8.8   -4.4      F          NaN   0    0
34          @  MEM  ...   2   1   13.0   7.2   -4.0      F          NaN   0    0
35          @  ORL  ...   0   2   15.0  12.0    2.6      F          NaN   0    0
36          @  CHO  ...   0   3   13.0  10.2    2.5      F          NaN   0    0
37        NaN  ORL  ...   1   2    NaN  15.8    5.7      F          NaN   0    0
38        NaN  CHI  ...   0   1    NaN  14.6   18.4      F          NaN   0    0
39          @  UTA  ...   1   1   22.0  20.4    5.9      F          NaN   1    0
40          @  GSW  ...   3   1   10.0   3.6  -11.6      F          NaN   0    0
41        NaN  OKC  ...   2   0   11.0  10.1   -3.9      F          NaN   1    0
42          @  POR  ...   1   4   26.0  20.3    5.1      F          NaN   0    0
43        NaN  LAL  ...   0   1    6.0   8.8   -1.8      F          NaN   0    0
44        NaN  UTA  ...   2   0    3.0  -0.5   -9.9      F          NaN   0    0
45          @  LAL  ...   1   3   18.0  14.1    2.7      F          NaN   0    0
46          @  OKC  ...   0   0   27.0  26.9   12.8      F          NaN   0    0
47        NaN  SAS  ...   0   1    8.0   8.2   -3.1      F          NaN   0    0
48          @  SAS  ...   0   1    9.0  16.0    8.4      F          NaN   0    0
49          @  IND  ...   5   2   18.0  20.1    4.9      F          NaN   1    0

       FP
0    1.50
1    9.25
2    0.00
3    1.25
4    0.00
5   13.25
6   13.50
7    0.00
8    3.50
9   11.25
10   5.00
11   3.25
12  13.00
13  11.00
14   2.50
15   6.00
16   6.50
17  36.50
18   7.25
19   8.50
20  10.25
21   9.50
22  11.25
23  29.25
24  26.50
25  20.75
26  18.25
27  13.50
28   7.00
29  34.25
30  25.50
31  14.75
32  15.00
33  24.50
34  19.25
35  21.00
36  20.00
```

```
37  33.25
38  25.75
39  42.00
40  15.25
41  30.50
42  34.00
43  28.75
44   6.75
45  30.00
46  41.25
47  18.50
48  32.75
49  48.00

[50 rows x 42 columns]
```

In [9]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10179 entries, 0 to 10178
Data columns (total 42 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0  10179 non-null  int64
 1   Rk          10179 non-null  int64
 2   Player      10179 non-null  object
 3   PTS         10179 non-null  int64
 4   PTS.1       10179 non-null  int64
 5   Date        10179 non-null  datetime64[ns]
 6   Age         10179 non-null  object
 7   Team        10179 non-null  object
 8   Unnamed: 7  4681 non-null   object
 9   Opp         10179 non-null  object
 10  Result      10179 non-null  object
 11  GS          10179 non-null  int64
 12  MP          10179 non-null  int64
 13  FG          10179 non-null  int64
 14  FGA         10179 non-null  int64
 15  FG%         9681 non-null   float64
 16  2P          10179 non-null  int64
 17  2PA         10179 non-null  int64
 18  2P%         9015 non-null   float64
 19  3P          10179 non-null  int64
 20  3PA         10179 non-null  int64
 21  3P%         7966 non-null   float64
 22  FT          10179 non-null  int64
 23  FTA         10179 non-null  int64
 24  FT%         5718 non-null   float64
 25  TS%         9753 non-null   float64
 26  ORB         10179 non-null  int64
 27  DRB         10179 non-null  int64
 28  TRB         10179 non-null  int64
 29  AST         10179 non-null  int64
 30  STL         10179 non-null  int64
 31  BLK         10179 non-null  int64
 32  TOV         10179 non-null  int64
 33  PF          10179 non-null  int64
 34  PTS.2       9335 non-null   float64
 35  GmSc        10179 non-null  float64
 36  BPM         10179 non-null  float64
 37  Pos.        10179 non-null  object
 38  Unnamed: 6  424 non-null    object
 39  DD          10179 non-null  int32
 40  TD          10179 non-null  int32
 41  FP          10179 non-null  float64
```

```
dtypes: datetime64[ns](1), float64(9), int32(2), int64(22), object(8)
memory usage: 3.2+ MB
```

In [10]:
```
#Return Age column after we sort out the data type
df =df.drop(columns=['Age','Player','Rk','Unnamed: 6','FT%','3P%','Unnamed: 0','2P%','PT
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10179 entries, 0 to 10178
Data columns (total 28 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PTS          10179 non-null  int64  
 1   Unnamed: 7   4681 non-null   object 
 2   GS           10179 non-null  int64  
 3   MP           10179 non-null  int64  
 4   FG           10179 non-null  int64  
 5   FGA          10179 non-null  int64  
 6   FG%          9681 non-null   float64
 7   2P           10179 non-null  int64  
 8   2PA          10179 non-null  int64  
 9   3P           10179 non-null  int64  
 10  3PA          10179 non-null  int64  
 11  FT           10179 non-null  int64  
 12  FTA          10179 non-null  int64  
 13  TS%          9753 non-null   float64
 14  ORB          10179 non-null  int64  
 15  DRB          10179 non-null  int64  
 16  TRB          10179 non-null  int64  
 17  AST          10179 non-null  int64  
 18  STL          10179 non-null  int64  
 19  BLK          10179 non-null  int64  
 20  TOV          10179 non-null  int64  
 21  PF           10179 non-null  int64  
 22  GmSc         10179 non-null  float64
 23  BPM          10179 non-null  float64
 24  Pos.         10179 non-null  object 
 25  DD           10179 non-null  int32  
 26  TD           10179 non-null  int32  
 27  FP           10179 non-null  float64
dtypes: float64(5), int32(2), int64(19), object(2)
memory usage: 2.1+ MB
```

In [11]:
```
#Checking for missing values
df.isnull().sum()
```

Out[11]:
```
PTS              0
Unnamed: 7    5498
GS               0
MP               0
FG               0
FGA              0
FG%            498
2P               0
2PA              0
3P               0
3PA              0
FT               0
FTA              0
TS%            426
ORB              0
DRB              0
TRB              0
AST              0
STL              0
BLK              0
```

```
TOV              0
PF               0
GmSc             0
BPM              0
Pos.             0
DD               0
TD               0
FP               0
dtype: int64
```

In [12]:

```
df['Unnamed: 7'] = df['Unnamed: 7'].map({'NaN':'0','@': '1'})
df.head(50)
#Median of the Age column
#print('Median of FG% column: %.2f' % (df["FG%"].median(skipna = True)))

#Median of the Age column
#print('Median of Age column: %.2f' % (dataset["Age"].median(skipna = True)))

#Percentage of missing records in the Cabin column
#print('Percent of missing records in the Cabin column: %.2f%%' %((dataset['Cabin'].isnu

#Most common boarding port of embarkation
#print('Most common boarding port of embarkation: %s' %dataset['Embarked'].value_counts(
```

Out[12]:

	PTS	Unnamed: 7	GS	MP	FG	FGA	FG%	2P	2PA	3P	...	STL	BLK	TOV	PF	GmSc	BPM	Pos.	DD	T
0	0	NaN	0	2	0	0	NaN	0	0	0	...	0	0	0	0	0.7	10.3	G	0	
1	3	NaN	0	15	1	4	0.250	0	0	1	...	1	0	0	2	2.1	-4.3	G	0	
2	0	NaN	0	1	0	0	NaN	0	0	0	...	0	0	0	0	0.0	-8.1	G	0	
3	0	NaN	0	3	0	0	NaN	0	0	0	...	0	0	0	1	-0.1	-9.0	G	0	
4	0	NaN	0	2	0	1	0.000	0	0	0	...	0	0	0	1	-1.1	-41.6	G	0	
5	8	1	0	14	2	3	0.667	0	1	2	...	0	0	1	1	6.9	7.6	G	0	
6	12	1	0	10	4	6	0.667	1	1	3	...	0	0	0	2	8.6	17.4	G	0	
7	0	NaN	0	1	0	0	NaN	0	0	0	...	0	0	0	0	0.0	-9.2	G	0	
8	2	NaN	0	3	1	2	0.500	1	1	0	...	0	0	0	1	1.3	-3.1	G	0	
9	10	1	0	14	4	6	0.667	2	2	2	...	0	0	1	1	6.7	2.4	G	0	
10	3	NaN	0	4	1	2	0.500	0	0	1	...	0	0	0	2	1.9	6.7	G	0	
11	2	1	0	2	1	1	1.000	1	1	0	...	0	0	0	1	1.6	4.8	G	0	
12	8	NaN	0	6	3	4	0.750	1	1	2	...	2	0	0	0	8.4	42.7	F	0	
13	10	1	0	6	4	6	0.667	2	3	2	...	0	0	0	2	6.6	16.8	F	0	
14	0	1	0	5	0	2	0.000	0	2	0	...	0	0	0	0	0.0	-13.7	F	0	
15	2	1	0	14	1	3	0.333	1	1	0	...	0	0	0	1	1.2	-8.6	F	0	
16	3	1	0	9	1	2	0.500	0	0	1	...	0	0	0	0	3.4	5.1	F	0	
17	24	NaN	0	31	10	15	0.667	8	9	2	...	3	0	1	2	21.0	11.9	F	0	
18	3	NaN	0	10	1	5	0.200	0	1	1	...	0	0	0	1	0.8	-10.9	F	0	
19	3	NaN	0	14	1	3	0.333	0	1	1	...	1	0	1	2	1.8	0.0	F	0	
20	9	1	0	12	3	7	0.429	1	5	2	...	0	0	1	0	4.6	2.1	F	0	
21	7	1	0	17	3	6	0.500	2	4	1	...	0	0	2	1	3.3	-3.8	F	0	

22	6	NaN	0	21	3	9	0.333	3	5	0	...	1	0	2	1	2.3	-10.0	F	0
23	17	NaN	0	30	8	15	0.533	7	9	1	...	1	1	0	2	14.0	4.0	F	0
24	17	1	1	36	7	11	0.636	4	5	3	...	3	0	2	3	13.6	4.9	F	0
25	12	NaN	0	24	4	8	0.500	2	4	2	...	1	0	1	0	10.7	5.3	F	0
26	11	1	0	24	3	5	0.600	0	1	3	...	2	0	1	2	9.9	5.6	F	0
27	8	NaN	0	21	3	13	0.231	1	4	2	...	1	0	0	1	1.3	-10.7	F	0
28	3	1	0	16	1	6	0.167	0	0	1	...	1	0	1	2	-0.6	-13.8	F	0
29	24	NaN	1	35	11	16	0.688	9	11	2	...	3	0	2	0	20.2	7.1	F	0
30	11	NaN	1	30	4	15	0.267	1	6	3	...	3	0	1	2	6.3	-1.8	F	0
31	9	1	1	29	3	13	0.231	2	5	1	...	1	0	1	2	2.0	-9.9	F	0
32	10	1	1	20	4	11	0.364	3	6	1	...	1	0	3	0	3.9	-9.4	F	0
33	17	NaN	1	40	7	17	0.412	5	6	2	...	0	1	1	2	8.8	-4.4	F	0
34	13	1	1	25	5	12	0.417	3	7	2	...	1	0	2	1	7.2	-4.0	F	0
35	15	1	1	27	6	9	0.667	3	5	3	...	0	1	0	2	12.0	2.6	F	0
36	13	1	0	28	5	8	0.625	2	2	3	...	0	0	0	3	10.2	2.5	F	0
37	19	NaN	0	25	8	13	0.615	5	9	3	...	1	0	1	2	15.8	5.7	F	0
38	14	NaN	0	21	5	7	0.714	1	1	4	...	3	0	0	1	14.6	18.4	F	0
39	22	1	1	33	10	17	0.588	10	15	0	...	1	1	1	1	20.4	5.9	F	1
40	10	1	1	29	5	12	0.417	5	6	0	...	0	0	3	1	3.6	-11.6	F	0
41	11	NaN	1	28	3	9	0.333	2	6	1	...	0	2	2	0	10.1	-3.9	F	1
42	26	1	1	28	12	16	0.750	12	14	0	...	0	0	1	4	20.3	5.1	F	0
43	6	NaN	1	29	2	8	0.250	2	5	0	...	0	2	0	1	8.8	-1.8	F	0
44	3	NaN	1	21	1	4	0.250	0	1	1	...	0	2	2	0	-0.5	-9.9	F	0
45	18	1	1	31	7	11	0.636	4	7	3	...	0	2	1	3	14.1	2.7	F	0
46	27	1	1	36	10	13	0.769	7	9	3	...	1	0	0	0	26.9	12.8	F	0
47	8	NaN	1	23	4	6	0.667	4	5	0	...	0	0	0	1	8.2	-3.1	F	0
48	9	1	1	30	3	5	0.600	3	4	0	...	4	1	0	1	16.0	8.4	F	0
49	18	1	1	34	5	8	0.625	4	5	1	...	1	1	5	2	20.1	4.9	F	1

50 rows × 28 columns

In [13]:

```
df["Unnamed: 7"].fillna(0, inplace=True)
df.head(50)
#Filling Age column by median
#dataset["Age"].fillna(dataset["Age"].median(skipna=True), inplace=True)
#Fillimg Embarked column by the most common port of embarkation
#dataset["Embarked"].fillna(dataset['Embarked'].value_counts().idxmax(), inplace=True)
#Dropping the cabin columns
#dataset.drop('Cabin', axis=1, inplace=True)
```

Out[13]:

	PTS	Unnamed: 7	GS	MP	FG	FGA	FG%	2P	2PA	3P	...	STL	BLK	TOV	PF	GmSc	BPM	Pos.	DD	T
0	0	0	0	2	0	0	NaN	0	0	0	...	0	0	0	0	0.7	10.3	G	0	

1	3	0	0	15	1	4	0.250	0	0	1	...	1	0	0	2	2.1	-4.3	G	0
2	0	0	0	1	0	0	NaN	0	0	0	...	0	0	0	0	0.0	-8.1	G	0
3	0	0	0	3	0	0	NaN	0	0	0	...	0	0	0	1	-0.1	-9.0	G	0
4	0	0	0	2	0	1	0.000	0	0	0	...	0	0	0	1	-1.1	-41.6	G	0
5	8	1	0	14	2	3	0.667	0	1	2	...	0	0	1	1	6.9	7.6	G	0
6	12	1	0	10	4	6	0.667	1	1	3	...	0	0	0	2	8.6	17.4	G	0
7	0	0	0	1	0	0	NaN	0	0	0	...	0	0	0	0	0.0	-9.2	G	0
8	2	0	0	3	1	2	0.500	1	1	0	...	0	0	0	1	1.3	-3.1	G	0
9	10	1	0	14	4	6	0.667	2	2	2	...	0	0	1	1	6.7	2.4	G	0
10	3	0	0	4	1	2	0.500	0	0	1	...	0	0	0	2	1.9	6.7	G	0
11	2	1	0	2	1	1	1.000	1	1	0	...	0	0	0	1	1.6	4.8	G	0
12	8	0	0	6	3	4	0.750	1	1	2	...	2	0	0	0	8.4	42.7	F	0
13	10	1	0	6	4	6	0.667	2	3	2	...	0	0	0	2	6.6	16.8	F	0
14	0	1	0	5	0	2	0.000	0	2	0	...	0	0	0	0	0.0	-13.7	F	0
15	2	1	0	14	1	3	0.333	1	1	0	...	0	0	0	1	1.2	-8.6	F	0
16	3	1	0	9	1	2	0.500	0	0	1	...	0	0	0	0	3.4	5.1	F	0
17	24	0	0	31	10	15	0.667	8	9	2	...	3	0	1	2	21.0	11.9	F	0
18	3	0	0	10	1	5	0.200	0	1	1	...	0	0	0	1	0.8	-10.9	F	0
19	3	0	0	14	1	3	0.333	0	1	1	...	1	0	1	2	1.8	0.0	F	0
20	9	1	0	12	3	7	0.429	1	5	2	...	0	0	1	0	4.6	2.1	F	0
21	7	1	0	17	3	6	0.500	2	4	1	...	0	0	2	1	3.3	-3.8	F	0
22	6	0	0	21	3	9	0.333	3	5	0	...	1	0	2	1	2.3	-10.0	F	0
23	17	0	0	30	8	15	0.533	7	9	1	...	1	1	0	2	14.0	4.0	F	0
24	17	1	1	36	7	11	0.636	4	5	3	...	3	0	2	3	13.6	4.9	F	0
25	12	0	0	24	4	8	0.500	2	4	2	...	1	0	1	0	10.7	5.3	F	0
26	11	1	0	24	3	5	0.600	0	1	3	...	2	0	1	2	9.9	5.6	F	0
27	8	0	0	21	3	13	0.231	1	4	2	...	1	0	0	1	1.3	-10.7	F	0
28	3	1	0	16	1	6	0.167	0	0	1	...	1	0	1	2	-0.6	-13.8	F	0
29	24	0	1	35	11	16	0.688	9	11	2	...	3	0	2	0	20.2	7.1	F	0
30	11	0	1	30	4	15	0.267	1	6	3	...	3	0	1	2	6.3	-1.8	F	0
31	9	1	1	29	3	13	0.231	2	5	1	...	1	0	1	2	2.0	-9.9	F	0
32	10	1	1	20	4	11	0.364	3	6	1	...	1	0	3	0	3.9	-9.4	F	0
33	17	0	1	40	7	17	0.412	5	6	2	...	0	1	1	2	8.8	-4.4	F	0
34	13	1	1	25	5	12	0.417	3	7	2	...	1	0	2	1	7.2	-4.0	F	0
35	15	1	1	27	6	9	0.667	3	5	3	...	0	1	0	2	12.0	2.6	F	0
36	13	1	0	28	5	8	0.625	2	2	3	...	0	0	0	3	10.2	2.5	F	0
37	19	0	0	25	8	13	0.615	5	9	3	...	1	0	1	2	15.8	5.7	F	0

38	14	0	0	21	5	7	0.714	1	1	4	...	3	0	0	1	14.6	18.4	F	0
39	22	1	1	33	10	17	0.588	10	15	0	...	1	1	1	1	20.4	5.9	F	1
40	10	1	1	29	5	12	0.417	5	6	0	...	0	0	3	1	3.6	-11.6	F	0
41	11	0	1	28	3	9	0.333	2	6	1	...	0	2	2	0	10.1	-3.9	F	1
42	26	1	1	28	12	16	0.750	12	14	0	...	0	0	1	4	20.3	5.1	F	0
43	6	0	1	29	2	8	0.250	2	5	0	...	0	2	0	1	8.8	-1.8	F	0
44	3	0	1	21	1	4	0.250	0	1	1	...	0	2	2	0	-0.5	-9.9	F	0
45	18	1	1	31	7	11	0.636	4	7	3	...	0	2	1	3	14.1	2.7	F	0
46	27	1	1	36	10	13	0.769	7	9	3	...	1	0	0	0	26.9	12.8	F	0
47	8	0	1	23	4	6	0.667	4	5	0	...	0	0	0	1	8.2	-3.1	F	0
48	9	1	1	30	3	5	0.600	3	4	0	...	4	1	0	1	16.0	8.4	F	0
49	18	1	1	34	5	8	0.625	4	5	1	...	1	1	5	2	20.1	4.9	F	1

50 rows × 28 columns

In [14]:

```
#Separate the Age column after the at the - do this later
```

In [15]:

```
#Drop rows with NA
df = df.dropna()
#Checking for missing values
df.isnull().sum()
```

Out[15]:

```
PTS           0
Unnamed: 7    0
GS            0
MP            0
FG            0
FGA           0
FG%           0
2P            0
2PA           0
3P            0
3PA           0
FT            0
FTA           0
TS%           0
ORB           0
DRB           0
TRB           0
AST           0
STL           0
BLK           0
TOV           0
PF            0
GmSc          0
BPM           0
Pos.          0
DD            0
TD            0
FP            0
dtype: int64
```

In [16]:

```
#Import label encoder
from sklearn import preprocessing

#label_encoder object knows how to understand word labels
```

```
label_encoder = preprocessing.LabelEncoder()

#Encode labels in column Sex and Embarked
#df['Unnamed: 7']= label_encoder.fit_transform(df['Unnamed: 7'])
df['Pos.']=label_encoder.fit_transform(df['Pos.'])
```

In [17]:
```
df.head()
```

Out[17]:

	PTS	Unnamed: 7	GS	MP	FG	FGA	FG%	2P	2PA	3P	...	STL	BLK	TOV	PF	GmSc	BPM	Pos.	DD	TD
1	3	0	0	15	1	4	0.250	0	0	1	...	1	0	0	2	2.1	-4.3	5	0	0
4	0	0	0	2	0	1	0.000	0	0	0	...	0	0	0	1	-1.1	-41.6	5	0	0
5	8	1	0	14	2	3	0.667	0	1	2	...	0	0	1	1	6.9	7.6	5	0	0
6	12	1	0	10	4	6	0.667	1	1	3	...	0	0	0	2	8.6	17.4	5	0	0
8	2	0	0	3	1	2	0.500	1	1	0	...	0	0	0	1	1.3	-3.1	5	0	0

5 rows × 28 columns

In [18]:
```
#df['Age'] = df['Age'].astype(str).astype(float)
df['Unnamed: 7'] = df['Unnamed: 7'].astype(str).astype(float)
df['Pos.'] = df['Pos.'].astype(str).astype(float)
```

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9681 entries, 1 to 10178
Data columns (total 28 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   PTS         9681 non-null   int64  
 1   Unnamed: 7  9681 non-null   float64
 2   GS          9681 non-null   int64  
 3   MP          9681 non-null   int64  
 4   FG          9681 non-null   int64  
 5   FGA         9681 non-null   int64  
 6   FG%         9681 non-null   float64
 7   2P          9681 non-null   int64  
 8   2PA         9681 non-null   int64  
 9   3P          9681 non-null   int64  
 10  3PA         9681 non-null   int64  
 11  FT          9681 non-null   int64  
 12  FTA         9681 non-null   int64  
 13  TS%         9681 non-null   float64
 14  ORB         9681 non-null   int64  
 15  DRB         9681 non-null   int64  
 16  TRB         9681 non-null   int64  
 17  AST         9681 non-null   int64  
 18  STL         9681 non-null   int64  
 19  BLK         9681 non-null   int64  
 20  TOV         9681 non-null   int64  
 21  PF          9681 non-null   int64  
 22  GmSc        9681 non-null   float64
 23  BPM         9681 non-null   float64
 24  Pos.        9681 non-null   float64
 25  DD          9681 non-null   int32  
 26  TD          9681 non-null   int32  
 27  FP          9681 non-null   float64
dtypes: float64(7), int32(2), int64(19)
memory usage: 2.1 MB
```

In [20]:
```
df.shape
```
Out[20]:
```
(9681, 28)
```

In [21]:
```
df.corr()
```
Out[21]:

	PTS	Unnamed: 7	GS	MP	FG	FGA	FG%	2P	2PA	
PTS	1.000000	-0.022043	0.515122	0.717239	0.965029	0.884617	0.317897	0.810541	0.789690	0.626
Unnamed: 7	-0.022043	1.000000	-0.007563	-0.009890	-0.020686	-0.007005	-0.015295	-0.010361	0.002094	-0.026
GS	0.515122	-0.007563	1.000000	0.702819	0.509195	0.563009	0.058587	0.463787	0.523943	0.265
MP	0.717239	-0.009890	0.702819	1.000000	0.698279	0.765422	0.099132	0.591655	0.658613	0.444
FG	0.965029	-0.020686	0.509195	0.698279	1.000000	0.882392	0.373528	0.889194	0.829078	0.560
FGA	0.884617	-0.007005	0.563009	0.765422	0.882392	1.000000	0.035483	0.759791	0.874629	0.539
FG%	0.317897	-0.015295	0.058587	0.099132	0.373528	0.035483	1.000000	0.359202	0.120395	0.160
2P	0.810541	-0.010361	0.463787	0.591655	0.889194	0.759791	0.359202	1.000000	0.896498	0.118
2PA	0.789690	0.002094	0.523943	0.658613	0.829078	0.874629	0.120395	0.896498	1.000000	0.175
3P	0.626533	-0.026130	0.265207	0.444005	0.560002	0.539097	0.160171	0.118890	0.175851	1.000
3PA	0.596814	-0.017002	0.348441	0.552681	0.534465	0.701219	-0.106533	0.191651	0.267676	0.812
FT	0.675903	-0.009047	0.351209	0.472086	0.500377	0.532363	0.064738	0.502657	0.550577	0.175
FTA	0.666155	-0.010538	0.357668	0.477213	0.507515	0.530622	0.080683	0.522780	0.565005	0.154
TS%	0.372506	-0.016507	0.063487	0.141865	0.368837	0.064388	0.921559	0.255200	0.061387	0.338
ORB	0.160110	-0.003195	0.186446	0.225316	0.192364	0.159849	0.112917	0.287914	0.297870	-0.103
DRB	0.423495	-0.020550	0.400581	0.531101	0.430365	0.406065	0.147307	0.452201	0.451438	0.115
TRB	0.400099	-0.017584	0.392457	0.511574	0.418458	0.386165	0.162070	0.474022	0.477401	0.049
AST	0.454600	-0.020492	0.404434	0.536408	0.432663	0.506299	0.002145	0.381234	0.451597	0.248
STL	0.263450	-0.004203	0.230302	0.333253	0.259949	0.282824	0.030270	0.227744	0.247881	0.151
BLK	0.136289	-0.009406	0.153096	0.192701	0.150210	0.109886	0.106898	0.191502	0.171463	-0.020
TOV	0.455901	0.007067	0.373239	0.466341	0.433949	0.465917	0.052981	0.402782	0.442265	0.212
PF	0.249074	0.001102	0.301803	0.369995	0.248479	0.232793	0.112808	0.238278	0.238418	0.107
GmSc	0.923538	-0.034509	0.468557	0.662234	0.896681	0.727590	0.426754	0.773034	0.687960	0.546
BPM	0.479013	-0.019189	0.129727	0.237118	0.466746	0.248343	0.622090	0.346247	0.220049	0.385
Pos.	0.092910	-0.009412	0.007623	0.100556	0.058978	0.172558	-0.193374	-0.049306	0.028969	0.217
DD	0.301568	-0.008403	0.228756	0.279904	0.311735	0.277488	0.101404	0.357988	0.343481	0.028
TD	0.088630	0.003099	0.054135	0.076680	0.075766	0.087909	0.001255	0.078316	0.088171	0.022
FP	0.901709	-0.028172	0.572939	0.787692	0.879802	0.834016	0.265854	0.769724	0.775948	0.515

28 rows × 28 columns

In [22]:
```
from sklearn.model_selection import train_test_split
```

```
X = df.drop('FP', axis=1)  # Features
y = df['FP']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42
```

In [23]:

```
# Create a multiple linear regression model
linear_regression = LinearRegression()

# Create the RFE model and select 3 attributes
rfe = RFE(linear_regression,n_features_to_select=3)
rfe = rfe.fit(X_train, y_train)

# Print the features that were selected
print(X_train.columns[rfe.support_])
X_train = X_train[['AST', 'TRB', 'PTS','3P']]
X_train
```

```
Index(['AST', 'BLK', 'GmSc'], dtype='object')
```

Out[23]:

	AST	TRB	PTS	3P
6677	1	8	15	1
6767	1	2	14	4
9580	3	2	11	0
2224	5	9	28	1
7811	1	6	15	1
...	...	...	...	...
5989	3	6	27	3
5407	3	7	26	4
5630	5	8	15	1
906	3	2	29	0
7617	0	1	0	0

6776 rows × 4 columns

In [24]:

```
X_test = X_test[['AST', 'TRB', 'PTS','3P']]
```

In [25]:

```
# Fit the model on the training data
linear_regression.fit(X_train, y_train)

# Make predictions on the testing data
predictions = linear_regression.predict(X_test)

# Evaluate the model's performance
from sklearn.metrics import mean_absolute_error, mean_squared_error,r2_score

print('MAE:', mean_absolute_error(y_test, predictions))
print('MSE:', mean_squared_error(y_test, predictions))
print(r2_score(y_test, predictions))
```

```
MAE: 1.9839174817980219
MSE: 6.842806248434573
0.9676715079903068
```

In [26]:

```
df = X_test
df['FP_Predicted'] = predictions
```

In [27]:
```
df
```

Out[27]:

	AST	TRB	PTS	3P	FP_Predicted
7251	0	4	2	0	8.199856
7335	1	0	3	0	4.993254
5959	1	0	3	1	5.410367
4641	2	4	4	0	13.141454
7746	1	3	2	0	8.242286
...	...	...	...	...	...
5019	1	3	4	0	10.260324
5600	0	0	2	0	2.522455
8701	1	3	10	0	16.314436
9387	0	0	6	2	7.392755
5659	1	8	19	1	32.909468

2905 rows × 5 columns

In [ ]: