UNIVERSIDADE DE FORTALEZA

MBA Ciência de Dados

Nome: DiegoTeixeira Marques
E-mail: diegoteixeira1996@gmail.com
Matrícula: 1924526

1. Introdução

Com o objetivo de praticar o que foi visto em sala, este trabalho visa desenvolver um modelo de Machine Learning capaz de fazer previsões. Os dados utilizados são de jogadores da NBA, cujas informações contidas são referentes a seus rendimentos em quadra. Assim, o desafio é chegar num modelo capaz de prever os salários dos jogadores, verificando também, se seu ganho é justo em relação aos demais.

2. Metodologia

Os primeiros passos, após a importação dos módulos e carga dos dados, será analisar todo o DataFrame e fazer uma higienização nos valores que não têm significância para as análises. Será verificado, os dados nulos, valores em branco e/ou zerados, os tipos de dados e se há informações repetidas

Em seguida, será analisada as correlações existentes do "SALARIO" (variável alvo) com os demais atributos através de gráficos e números aprasentados a partir do módulo SEABORN e da fórmula de PEARSON.

Por fim, será criado um modelo de Regressão Linear com os atribudos de maior correlação. Os testes e validações também ocorrerão, assim como os ajustes em prol de um modelo mais apropriado para previsões.

In [271]:

```
# Importando os módulos necessários

import pandas as pd
import seaborn as sb
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

In [272]:

```
# Carregando o DATAFRAME

df_nba = pd.read_csv('NBA_Players.csv')
```

3. Higienização do DataFrame

3.1 Conhecendo os dados

Os comandos a seguir permitem ter uma análise geral dos dados e identificar pontos que venham dificultar a continuidade do desenvolvimento do modelo.

In [273]:

```
# Visualizando os dados

df_nba.head(5)
```

Out[273]:

	TEAM	NAME	EXPERIENCE	URL	POSITION	AGE	
0	Boston Celtics	Aron Baynes	6	http://www.espn.com/nba/player/_/id/2968439	SF	31	208
1	Boston Celtics	Justin Bibbs	0	http://www.espn.com/nba/player/_/id/3147500	G	22	195
2	Boston Celtics	Jabari Bird	1	http://www.espn.com/nba/player/_/id/3064308	SG	24	198
3	Boston Celtics	Jaylen Brown	2	http://www.espn.com/nba/player/_/id/3917376	F	21	200
4	Boston Celtics	PJ Dozier	1	http://www.espn.com/nba/player/_/id/3923250	PG	21	198

5 rows × 30 columns

In [274]:

```
# Verificando quantidade de Linhas e Colunas

df_nba.shape
```

Out[274]:

```
(550, 30)
```

In [275]:

```
# Visualizando as colunas
# Verifica-se um espaço desnecessário no nome da coluna

df_nba.columns
```

Out[275]:

```
Index(['TEAM', ' NAME', ' EXPERIENCE', ' URL', ' POSITION', ' AGE', ' HT',
       ' WT', ' COLLEGE', ' SALARY', ' PPG_LAST_SEASON', ' APG_LAST_SEASON',
       ' RPG_LAST_SEASON', ' PER_LAST_SEASON', ' PPG_CAREER', ' APG_CAREER',
       ' RGP_CAREER', ' GP', ' MPG', ' FGM_FGA', ' FGP', ' THM_THA', ' THP',
       ' FTM_FTA', ' FTP', ' APG', ' BLKPG', ' STLPG', ' TOPG', ' PPG'],
      dtype='object')
```

In [276]:

```
# Verificando os tipos dos dados
# Algumas colunas estão como object, embora apresente dados que normalmente são números, co
# Verificaremos cada dado dentro destas colunas para saber qual deles foge do contexto numé

df_nba.dtypes
```

Out[276]:

```
TEAM                object
 NAME               object
 EXPERIENCE          int64
 URL                object
 POSITION           object
 AGE                object
 HT                float64
 WT                float64
 COLLEGE            object
 SALARY             object
 PPG_LAST_SEASON   float64
 APG_LAST_SEASON   float64
 RPG_LAST_SEASON   float64
 PER_LAST_SEASON   float64
 PPG_CAREER        float64
 APG_CAREER        float64
 RGP_CAREER        float64
 GP                  int64
 MPG               float64
 FGM_FGA            object
 FGP               float64
 THM_THA            object
 THP               float64
 FTM_FTA            object
 FTP               float64
 APG               float64
 BLKPG             float64
 STLPG             float64
 TOPG              float64
 PPG               float64
dtype: object
```

In [277]:

```
# Identificando onde há valores nulos
# Observa-se 12 linhas com valores nulos. Serão analisadas e possivelmente excluídas do moa

df_nba.isnull().sum()
```

Out[277]:

```
TEAM                0
 NAME               0
 EXPERIENCE         0
 URL                0
 POSITION           0
 AGE                0
 HT                 0
 WT                 0
 COLLEGE            0
 SALARY             0
 PPG_LAST_SEASON    12
 APG_LAST_SEASON    12
 RPG_LAST_SEASON    12
 PER_LAST_SEASON    12
 PPG_CAREER         0
 APG_CAREER         0
 RGP_CAREER         0
 GP                 0
 MPG                0
 FGM_FGA            0
 FGP                0
 THM_THA            0
 THP                0
 FTM_FTA            0
 FTP                0
 APG                0
 BLKPG              0
 STLPG              0
 TOPG               0
 PPG                0
dtype: int64
```

In [278]:

```
# Verificando informações das colunas que são do tipo Object
# Apenas AGE e SALARY será usada para o modelo
# TEAM, NAME, URL e POSITION não apresentam valor número, por isso, serão desconsiderados,
# FGM_FGA, THM_THA, FTM_FTA são uma relação de tentativa e acerto, por exemplo, lances livr
    # Há colunas que representam esta relação em formato númerico: FGP, THP, FTP. Estas ser

df_nba.select_dtypes('object').head(5)
```

Out[278]:

	TEAM	NAME	URL	POSITION	AGE	COLLEGE	SAL
0	Boston Celtics	Aron Baynes	http://www.espn.com/nba/player/_/id/2968439	SF	31	Washington State	5,193
1	Boston Celtics	Justin Bibbs	http://www.espn.com/nba/player/_/id/3147500	G	22	Virginia Tech	sig
2	Boston Celtics	Jabari Bird	http://www.espn.com/nba/player/_/id/3064308	SG	24	California	1,349
3	Boston Celtics	Jaylen Brown	http://www.espn.com/nba/player/_/id/3917376	F	21	California	5,169
4	Boston Celtics	PJ Dozier	http://www.espn.com/nba/player/_/id/3923250	PG	21	South Carolina	sig

In [279]:

```
# Analisando os dados únicos das colunas do tipo Object que serão consideradas no modelo
# Será retirado as idades com '-'
# Será retirado os salários com 'Not signed'

idades = df_nba[' AGE'].unique()
salarios = df_nba[' SALARY'].unique()

print(idades, '\n', '\n', salarios)
```

```
['31' '22' '24' '21' '28' '32' '26' '23' '29' '20' '25' '33' '27' '19' '-'
 '30' '34' '18' '36' '37' '35' '40' '38' '41']

 ['5,193,600' 'Not signed' '1,349,464' '5,169,960' '31,214,295'
 '28,928,709' '20,099,189' '5,375,000' '1,378,242' '3,050,390'
 '11,660,716' '6,700,800' '838,464' '2,667,600' '2,034,120' '15,400,000'
 '18,500,000' '4,449,000' '1,656,092' '9,530,000' '13,764,045' '1,512,601'
 '8,000,000' '2,470,357' '1,618,320' '1,702,800' '1,632,240' '1,942,422'
 '7,019,698' '5,000,000' '4,544,000' '1,795,015' '17,325,000' '6,500,000'
 '18,622,514' '3,739,920' '1,619,260' '12,253,780' '4,294,480' '4,155,720'
 '5,697,054' '1,485,440' '7,119,650' '8,575,916' '12,800,562' '10,464,092'
 '25,467,250' '8,339,880' '1,740,000' '1,600,520' '12,250,000' '2,526,840'
 '1,703,649' '6,434,520' '10,000,000' '21,666,667' '23,114,067'
 '31,200,000' '8,333,333' '1,826,300' '9,367,200' '1,569,360' '1,544,951'
 '16,539,326' '8,653,847' '2,536,898' '5,337,000' '37,457,154'
 '30,000,000' '1,644,240' '17,469,565' '16,000,000' '8,307,692'
 '18,988,725' '5,027,028' '12,000,000' '21,587,579' '3,375,360'
 '13,565,218' '6,000,000' '14,800,000' '6,134,520' '7,000,000' '4,320,500'
 '3,046,200' '6,300,000' '1,349,383' '7,461,960' '3,500,000' '1,000,000'
 '1,655,160' '5,757,120' '35,654,150' '1,689,840' '1,487,694' '9,000,000'
 '1,762,080' '20,421,546' '15,000,000' '7,464,912' '8,165,160' '4,661,280'
 '3,314,365' '3,552,960' '13,585,000' '3,258,539' '6,041,520' '949,000'
 '1,238,464' '11,750,000' '7,305,600' '4,696,875' '3,000,000' '5,470,920'
 '2,207,040' '3,844,760' '2,807,880' '8,739,500' '5,460,000' '11,692,308'
 '11,011,234' '11,286,516' '4,441,200' '4,221,000' '8,740,980' '4,384,616'
 '1,990,520' '19,500,000' '14,357,750' '4,536,120' '20,000,000'
 '3,263,294' '2,494,346' '2,280,600' '12,500,000' '2,760,095' '19,000,000'
 '3,472,887' '7,560,000' '24,119,025' '2,272,391' '2,775,000' '4,068,600'
 '14,720,000' '1,952,760' '2,500,000' '25,434,263' '1,857,480'
 '32,088,932' '17,043,478' '3,940,402' '3,275,280' '10,002,681'
 '4,075,000' '10,500,000' '12,400,000' '1,911,960' '7,945,000' '2,407,560'
 '7,333,333' '21,000,000' '2,659,800' '3,410,284' '13,964,045'
 '24,157,303' '1,641,000' '9,607,500' '2,481,000' '11,327,466' '3,382,000'
 '2,799,720' '13,000,000' '10,607,143' '2,534,280' '3,710,850'
 '24,107,258' '1,230,000' '6,560,640' '22,897,200' '9,631,250' '3,819,960'
 '15,293,104' '3,206,160' '1,621,415' '13,500,375' '35,650,150'
 '3,651,480' '14,631,250' '7,969,537' '8,641,000' '30,521,115' '7,666,667'
 '5,915,040' '5,285,394' '12,252,928' '25,976,111' '2,205,000' '8,808,685'
 '1,567,707' '22,347,015' '6,153,846' '2,487,000' '27,739,975' '3,125,000'
 '16,800,000' '10,087,200' '11,571,429' '2,947,320' '2,357,160'
 '1,667,160' '2,516,048' '18,089,887' '1,634,640' '2,299,080' '7,200,000'
 '2,250,960' '4,350,000' '13,766,421' '1,620,480' '5,356,440' '24,000,000'
 '17,000,000' '3,206,640' '988,464' '3,627,842' '7,488,372' '3,447,480'
 '14,087,500' '13,528,090' '2,955,840' '18,109,175' '6,270,000'
 '14,651,700' '19,245,370' '12,537,527' '11,550,000' '25,434,262'
 '3,448,926' '7,250,000' '4,865,040' '1,050,000' '21,590,909' '2,639,314'
 '4,969,080' '2,416,222' '12,750,000' '2,749,080' '15,944,154' '3,454,500'
 '8,600,000' '3,208,630' '26,011,913' '12,650,000' '3,129,187' '5,450,000'
 '19,169,800' '11,830,358' '1,773,840' '2,000,000' '16,517,857'
 '2,166,360' '24,605,181' '1,874,640' '3,364,249' '29,230,769' '3,499,800'
```

```
'12,917,808' '2,894,160' '20,445,779' '15,170,787' '14,000,000'
'2,444,053' '2,160,746' '4,750,000' '7,839,435' '5,455,236' '2,118,840'
'30,560,700' '1,757,429' '5,451,600' '15,500,000' '6,957,105' '3,628,920'
'2,795,000' '10,837,079' '10,595,506' '27,977,689' '25,759,766'
'11,111,111' '1,760,520' '17,868,853' '2,074,320' '1,679,520'
'11,536,515' '7,305,825' '9,600,000' '16,900,000' '23,241,573'
'13,045,455' '3,111,480' '2,150,000' '14,975,000' '5,250,000' '3,360,000']
```

In [280]:

```
# Verificando se a coluna NAME contém somente valores únicos
# Contando os nomes e comparando com a quantidade de linhas

len(df_nba[' NAME'].unique()) == df_nba.shape[0]
```

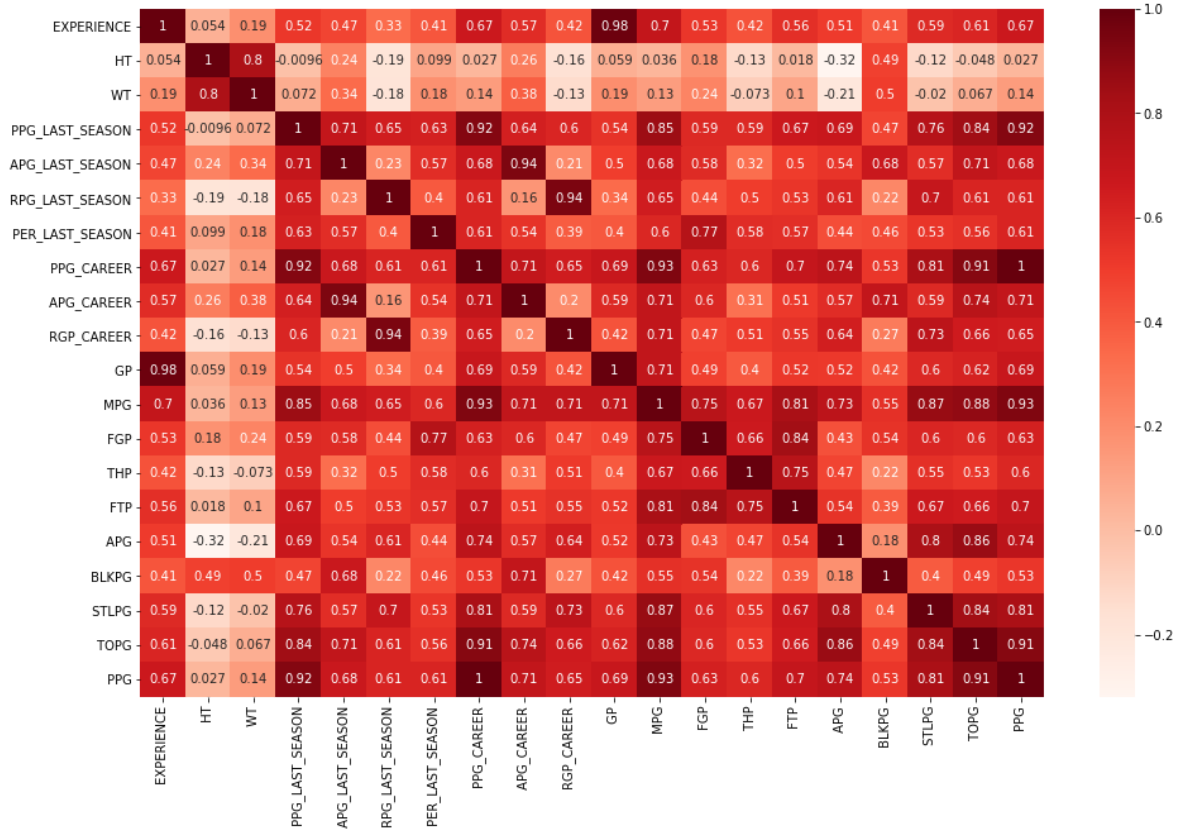
Out[280]:

```
True
```

In [281]:

```
# Analisando todas as correlações
# Verifica-se que PPG tem correlação 1 em relação a PPG_CAREER, portanto, são colunas com a
# PPG será desconsiderada

plt.figure(figsize=(16,10))
correlacao = df_nba.corr()
sb.heatmap(correlacao, annot=True, cmap=plt.cm.Reds)
plt.show()
```



3.2 Tratando os dados

Nesta etapa faremos as ações de limpeza referente os problemas vistos acima.

- Colunas com espaços em branco: '_NAME', '_EXPERIENCE', '_URL' ...
- Algumas colunas tem o tipo Object enquanto precisamos de um Int ou Float: 'AGE', 'THM_THA', 'FTM_FTA', 'FGM_FGA'
- Existem 12 linhas com valores nulos

In [282]:

```
# Criando uma cópia do DATAFRAME

df = df_nba.copy()
```

In [283]:

```
# Removendo espaços da descrição das colunas

df.columns = df_nba.columns.str.replace(' ', '')
df.columns
```

Out[283]:

```
Index(['TEAM', 'NAME', 'EXPERIENCE', 'URL', 'POSITION', 'AGE', 'HT', 'WT',
       'COLLEGE', 'SALARY', 'PPG_LAST_SEASON', 'APG_LAST_SEASON',
       'RPG_LAST_SEASON', 'PER_LAST_SEASON', 'PPG_CAREER', 'APG_CAREER',
       'RGP_CAREER', 'GP', 'MPG', 'FGM_FGA', 'FGP', 'THM_THA', 'THP',
       'FTM_FTA', 'FTP', 'APG', 'BLKPG', 'STLPG', 'TOPG', 'PPG'],
      dtype='object')
```

In [284]:

```
# Excluindo colunas que não entrarão para o modelo
# Optou-se por não rotular TEAM, POSITION, que pode conter uma correlação com SALARY
# Contudo esta correlação será vista mais adiante

df.drop(columns=['THM_THA', 'FTM_FTA', 'FGM_FGA', 'URL', 'TEAM', 'PPG', 'COLLEGE'], inplace
df.columns
```

Out[284]:

```
Index(['NAME', 'EXPERIENCE', 'POSITION', 'AGE', 'HT', 'WT', 'SALARY',
       'PPG_LAST_SEASON', 'APG_LAST_SEASON', 'RPG_LAST_SEASON',
       'PER_LAST_SEASON', 'PPG_CAREER', 'APG_CAREER', 'RGP_CAREER', 'GP',
       'MPG', 'FGP', 'THP', 'FTP', 'APG', 'BLKPG', 'STLPG', 'TOPG'],
      dtype='object')
```

In [285]:

```
#  Encontrando jogadores com valores nulos

df_nulos = df.filter(['PPG_LAST_SEASON', 'APG_LAST_SEASON', 'RPG_LAST_SEASON', 'PER_LAST_SE
df_nulos = df_nulos[df_nulos.PPG_LAST_SEASON == True]
df_nulos
```

Out[285]:

	PPG_LAST_SEASON	APG_LAST_SEASON	RPG_LAST_SEASON	PER_LAST_SEASON
50	True	True	True	True
257	True	True	True	True
276	True	True	True	True
309	True	True	True	True
332	True	True	True	True
335	True	True	True	True
347	True	True	True	True
448	True	True	True	True
473	True	True	True	True
482	True	True	True	True
484	True	True	True	True
514	True	True	True	True

In [286]:

```
# Extraindo os índices e verificando quem são os jogadores

indicesNulos = list(df_nulos.index)
df.loc[indicesNulos]
```

Out[286]:

	NAME	EXPERIENCE	POSITION	AGE	HT	WT	SALARY	PPG_LAST_SEASON
50	John Jenkins	5	SG	27	193.04	97.29	Not signed	NaN
257	CJ Wilcox	3	SG	27	195.58	88.24	Not signed	NaN
276	Christian Wood	2	PF	23	208.28	96.83	Not signed	NaN
309	Brandon Knight	6	PG	26	190.50	88.24	14,631,250	NaN
332	DJ Stephens	1	SG	27	195.58	85.07	Not signed	NaN
335	Alexis Ajinca	7	C	30	218.44	112.22	5,285,394	NaN
347	Darius Morris	4	PG	27	193.04	88.24	Not signed	NaN
448	Jordan McRae	2	SG	27	195.58	81.00	Not signed	NaN
473	Donald Sloan	5	PG	30	190.50	92.76	Not signed	NaN
482	Darius Johnson-Odom	2	SG	29	187.96	92.31	Not signed	NaN
484	James Nunnally	1	SF	28	200.66	94.12	1,349,383	NaN
514	Seth Curry	4	SG	28	187.96	83.71	2,795,000	NaN

12 rows × 23 columns

In [287]:

```
# Excluindo jogadores com informações nulas e conferindo se há mais alguma

df.drop(indicesNulos, axis=0, inplace=True)
df.isnull().sum()
```

Out[287]:

```
NAME               0
EXPERIENCE         0
POSITION           0
AGE                0
HT                 0
WT                 0
SALARY             0
PPG_LAST_SEASON    0
APG_LAST_SEASON    0
RPG_LAST_SEASON    0
PER_LAST_SEASON    0
PPG_CAREER         0
APG_CAREER         0
RGP_CAREER         0
GP                 0
MPG                0
FGP                0
THP                0
FTP                0
APG                0
BLKPG              0
STLPG              0
TOPG               0
dtype: int64
```

In [290]:

```
# Verificando quais jogadores não têm idade informada

df[df.AGE == '-']
```

Out[290]:

	NAME	EXPERIENCE	POSITION	AGE	HT	WT	SALARY	PPG_LAST_SEASON	APG
44	Phillip Carr	0	F	-	205.74	92.76	Not signed	0.0	
296	Tim Bond	0	G	-	198.12	76.92	Not signed	0.0	

2 rows × 23 columns

In [291]:

```
# Verificando quais jogadores não têm salário informado

df[df.SALARY == 'Not signed']
```

Out[291]:

	NAME	EXPERIENCE	POSITION	AGE	HT	WT	SALARY	PPG_LAST_SEASON
1	Justin Bibbs	0	G	22	195.58	99.55	Not signed	0.(
4	PJ Dozier	1	PG	21	198.12	92.76	Not signed	1.(
5	Marcus Georges-Hunt	2	SG	24	195.58	102.26	Not signed	1.4
9	Nick King	0	F	23	200.66	101.81	Not signed	0.(
10	Walt Lemon Jr.	1	PG	26	190.50	81.45	Not signed	3.4
...	...	...	...	...	...	...	...	..
533	Isaiah Cousins	0	PG	24	193.04	86.43	Not signed	0.(
538	Isaac Haas	0	C	22	218.44	131.22	Not signed	0.(
540	Trey Lewis	0	PG	25	187.96	83.71	Not signed	0.(
541	Jairus Lyles	0	PG	23	187.96	79.19	Not signed	0.(
543	Naz Mitrou-Long	1	SG	25	193.04	98.64	Not signed	3.(

102 rows × 23 columns

In [292]:

```
# Excluindo os jogadores sem salário informado garante também a exclusão dos que estão sem
# Faremos isso abaixo e ficaremos com 436 linhas e 23 colunas no novo DATAFRAME tratado

indicesColunasSemInformacao = list(df[df.SALARY == 'Not signed'].index)
df.drop(indicesColunasSemInformacao, axis=0, inplace=True)
df.shape
```

Out[292]:

```
(436, 23)
```

In [293]:

```
# Convertendo AGE e SALARY para float após retirar as linhas com valores do tipo object
# Convertendo também EXPERIENCE e GP que estavam como int64 para pradonizar os tipos numéri

df.SALARY = df.SALARY.str.replace(',', '').astype(float)
df.AGE = df.AGE.astype(float)

df.EXPERIENCE = df.EXPERIENCE.astype(float)
df.GP = df.GP.astype(float)

df.dtypes
```

Out[293]:

```
NAME               object
EXPERIENCE        float64
POSITION           object
AGE               float64
HT                float64
WT                float64
SALARY            float64
PPG_LAST_SEASON   float64
APG_LAST_SEASON   float64
RPG_LAST_SEASON   float64
PER_LAST_SEASON   float64
PPG_CAREER        float64
APG_CAREER        float64
RGP_CAREER        float64
GP                float64
MPG               float64
FGP               float64
THP               float64
FTP               float64
APG               float64
BLKPG             float64
STLPG             float64
TOPG              float64
dtype: object
```

4. Correlações

4.1 Visualizando os dados em gráficos

Os gráficos a seguir fazem parte da fase de encontrar correlações. Como a variável TARGET é a SALARY, vamos relacioná-la com todas as outras colunas. Será usado o SEABORN do python que nos fornece uma visualização prática e eficiente.

In [294]:

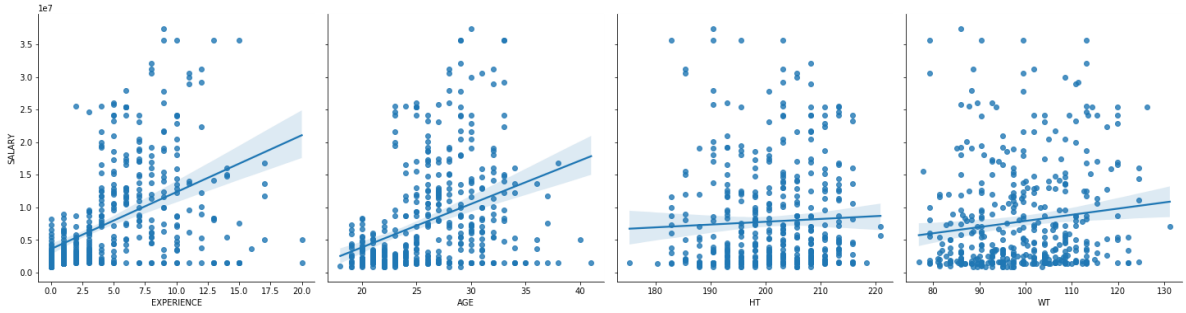
```
# Visualizando as correlações com SALARY

X = ['EXPERIENCE', 'AGE', 'HT', 'WT']
Y = ['SALARY']

sb.pairplot(df, x_vars=X, y_vars=Y, kind="reg", height=5)
```

Out[294]:

```
<seaborn.axisgrid.PairGrid at 0x17c941b0>
```



In [295]:

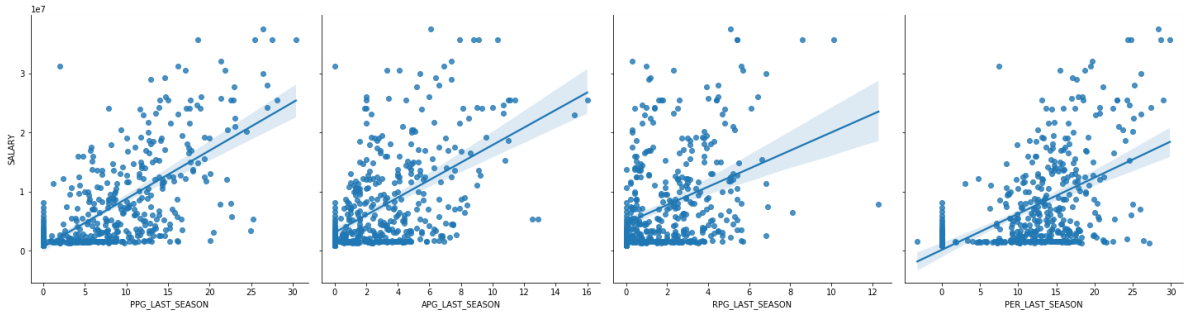
```
# Visualizando as correlações com SALARY

X = ['PPG_LAST_SEASON', 'APG_LAST_SEASON', 'RPG_LAST_SEASON', 'PER_LAST_SEASON']
Y = ['SALARY']

sb.pairplot(df, x_vars=X, y_vars=Y, kind="reg", height=5)
```

Out[295]:

```
<seaborn.axisgrid.PairGrid at 0x16485290>
```



In [296]:

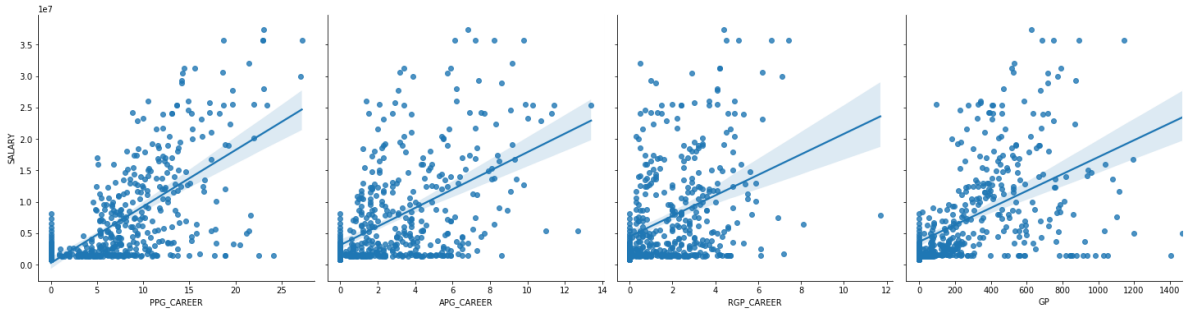
```
# Visualizando as correlações com SALARY

X = ['PPG_CAREER', 'APG_CAREER', 'RGP_CAREER', 'GP']
Y = ['SALARY']

sb.pairplot(df, x_vars=X, y_vars=Y, kind="reg", height=5)
```

Out[296]:

```
<seaborn.axisgrid.PairGrid at 0x178d8cf0>
```



In [297]:

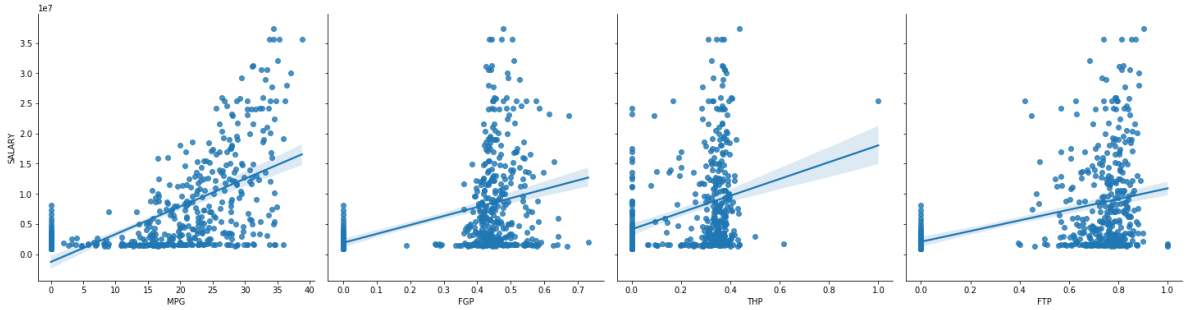
```
# Visualizando as correlações com SALARY

X = ['MPG', 'FGP', 'THP', 'FTP']
Y = ['SALARY']

sb.pairplot(df, x_vars=X, y_vars=Y, kind="reg", height=5)
```

Out[297]:

```
<seaborn.axisgrid.PairGrid at 0x17c74110>
```



In [298]:

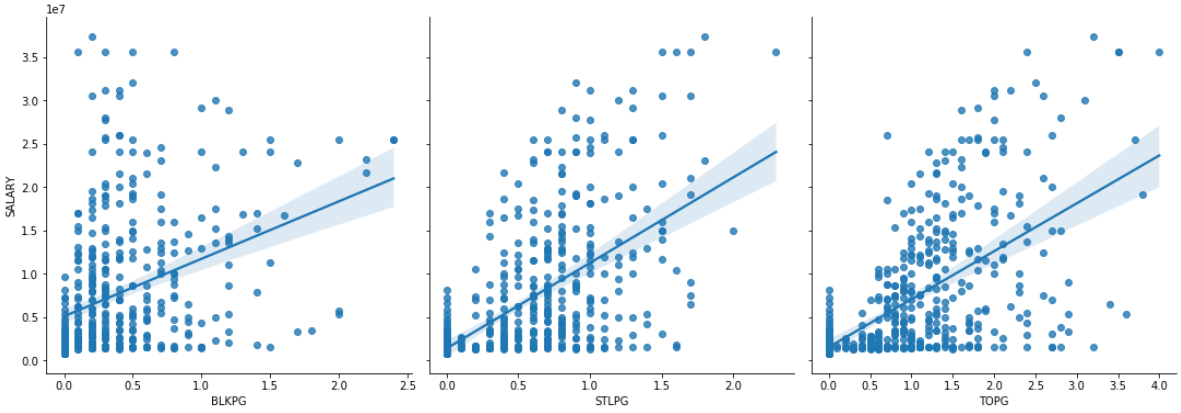
```
# Visualizando as correlações com SALARY

X = ['BLKPG', 'STLPG', 'TOPG']
Y = ['SALARY']

sb.pairplot(df, x_vars=X, y_vars=Y, kind="reg", height=5)
```

Out[298]:

```
<seaborn.axisgrid.PairGrid at 0x17aa08f0>
```



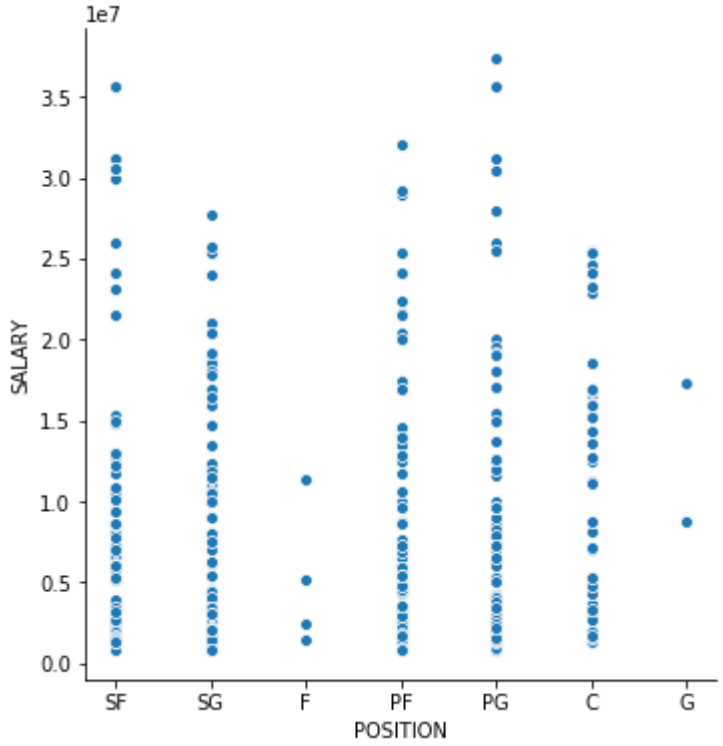
In [299]:

```
# Visualizando as correlações com SALARY
# Com exceção do F e G, as demais posições apresentam salários bem variados, limitando uma

sb.pairplot(df, x_vars=['POSITION'], y_vars=['SALARY'], height=5)
```

Out[299]:

```
<seaborn.axisgrid.PairGrid at 0x14df7cd0>
```



4.2 Analisando correlação em números (PEARSON)

Através da fórmula de PEARSON é possível descobrir o grau de correlação entre dois vetores de dados. Quanto mais próximo de 1 ou -1 maior a correlação. Para nossa análise, observa-se que a maior correlação atingiu 0.655 (PPG_LAST_SEASON), desconsideramos quando a análise é feita entre vetores de dados iguais.

In [300]:

```
# Verificando as correlações através da fórmula de PEARSON

correlacoes = []
df_correlacao = df.drop(columns=['NAME','POSITION'])
for i in df_correlacao.columns:
    correlacoes.append((df_correlacao['SALARY'].corr(df_correlacao[i]), i))

correlacoes.sort(reverse=True)
correlacoes
```

Out[300]:

```
[(0.9999999999999999, 'SALARY'),
 (0.655676397969779, 'PPG_LAST_SEASON'),
 (0.6372256737551588, 'PPG_CAREER'),
 (0.5849923238560933, 'MPG'),
 (0.5515693862593791, 'TOPG'),
 (0.5329677147941675, 'STLPG'),
 (0.5229449344237321, 'APG_LAST_SEASON'),
 (0.5047945352497611, 'PER_LAST_SEASON'),
 (0.4958263491929498, 'APG_CAREER'),
 (0.49027986931276785, 'GP'),
 (0.463271151100996, 'APG'),
 (0.45959857080706235, 'EXPERIENCE'),
 (0.3660154087928164, 'AGE'),
 (0.36047370428740527, 'RPG_LAST_SEASON'),
 (0.3562143720211878, 'RGP_CAREER'),
 (0.3523585623035294, 'BLKPG'),
 (0.2967581310929169, 'FGP'),
 (0.2931791234233112, 'FTP'),
 (0.2625040860060568, 'THP'),
 (0.12767084347920352, 'WT'),
 (0.04574213043579767, 'HT')]
```

5. Modelo de Machine Learning

5.1 Criando modelos

O modelo será baseado em regressão linear. Os dados serão divididos em teste e treinamento, em seguida será analisado o erro do modelo.

In [301]:

```
# Primeiro modelo: utilizaremos as 3 colunas com maior correlação

modelo = linear_model.LinearRegression()

# Target
y = df['SALARY']

# Colunas com maiores correlações
x = df[['PPG_LAST_SEASON', 'PPG_CAREER', 'MPG']]

# Dividindo os dados em 70% treino e 30% teste
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size=0.30, random_state=

# Treinando o modelo
modelo.fit(x_treino, y_treino)

# Aplicando a base de teste
previsao = modelo.predict(x_teste)

# Verificando erro
r2_score(y_teste,previsao)
```

Out[301]:

```
0.49100586670558277
```

In [302]:

```
# Segundo modelo: utilizaremos as 2 colunas com maior correlação e as 2 com menor

modelo = linear_model.LinearRegression()

# Target
y = df['SALARY']

# Colunas com maiores e menores correlações
x = df[['PPG_LAST_SEASON', 'PPG_CAREER', 'WT', 'HT']]

# Dividindo os dados em 70% treino e 30% teste
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size=0.30, random_state=

# Treinando o modelo
modelo.fit(x_treino, y_treino)

# Aplicando a base de teste
previsao = modelo.predict(x_teste)

# Verificando erro
r2_score(y_teste,previsao)
```

Out[302]:

```
0.5164612334369536
```

In [303]:

```
# Terceiro modelo: utilizaremos todas as colunas

modelo = linear_model.LinearRegression()

# Target
y = df['SALARY']

# Todas as colunas números
x = df[['EXPERIENCE', 'AGE', 'HT', 'WT', 'PPG_LAST_SEASON',
       'APG_LAST_SEASON', 'RPG_LAST_SEASON', 'PER_LAST_SEASON', 'PPG_CAREER', 'APG_CAREER',
       'RGP_CAREER', 'GP','MPG', 'FGP', 'THP', 'FTP', 'APG', 'BLKPG', 'STLPG', 'TOPG']]

# Dividindo os dados em 70% treino e 30% teste
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size=0.30, random_state=

# Treinando o modelo
modelo.fit(x_treino, y_treino)

# Aplicando a base de teste
previsao = modelo.predict(x_teste)

# Verificando erro
r2_score(y_teste,previsao)
```

Out[303]:

```
0.4981273218942057
```

In [304]:

```
# Quarto modelo: 5 colunas com maior correlação

modelo = linear_model.LinearRegression()

# Target
y = df['SALARY']

# 5 colunas números com maior correlação
x = df[['PPG_LAST_SEASON', 'PPG_CAREER', 'MPG', 'TOPG', 'STLPG']]

# Dividindo os dados em 70% treino e 30% teste
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size=0.30, random_state=

# Treinando o modelo
modelo.fit(x_treino, y_treino)

# Aplicando a base de teste
previsao = modelo.predict(x_teste)

# Verificando erro
r2_score(y_teste,previsao)
```

Out[304]:

```
0.5062617294839165
```

5.2 Escolhendo o modelo

Apesar de ser um erro ainda muito alto, o primeiro modelo apresentou um erro de 0.491, o menor dentre os analisados. Aplicaremos um caso prático a ele e tentaremos deduzir se o salário de um jogador é compatível ao que o modelo prever para ele.

In [305]:

```
# Primeiro modelo: utilizaremos as 3 colunas com maior correlação

modelo = linear_model.LinearRegression()

# Target
y = df['SALARY']

# Colunas com maiores correlações
x = df[['PPG_LAST_SEASON', 'PPG_CAREER', 'MPG']]

# Dividindo os dados em 70% treino e 30% teste
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size=0.30, random_state=

# Treinando o modelo
modelo.fit(x_treino, y_treino)
```

Out[305]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fal
se)
```

In [306]:

```
# Pegando informações de Stephen Curry para prever seu salário

stephen = df[df.NAME == 'Stephen Curry']
stephen = stephen.loc[:,['PPG_LAST_SEASON', 'PPG_CAREER', 'MPG']]
stephen
```

Out[306]:

	PPG_LAST_SEASON	PPG_CAREER	MPG
103	26.4	23.1	34.4

In [307]:

```
# Prevendo salário

previsao = round(modelo.predict(stephen)[0],0)
previsao
```

Out[307]:

```
20789870.0
```

In [308]:

```
# Pegando salário Real

salario = df[df.NAME == 'Stephen Curry'].SALARY
salario
```

Out[308]:

```
103    37457154.0
Name: SALARY, dtype: float64
```

In [309]:

```
# Erro de $ 16.667.284

ERRO = previsao - salario
ERRO
```

Out[309]:

```
103   -16667284.0
Name: SALARY, dtype: float64
```

In [310]:

```
# A previsão foi 44% menor do que o real

ErroPercente = round(ERRO / salario * 100,0)
ErroPercente
```

Out[310]:

```
103   -44.0
Name: SALARY, dtype: float64
```

6. Conclusão

O modelo ainda não apresenta números precisos em suas previsões. Contudo, com esta análise é possível deduzir que Stephen Curry recebe um valor acima do que seria ideal. Por fim, é preciso continuar o trabalho no modelo, aplicando um treinamento maior, analisando outras correlações, como a posição e o time que joga, por exemplo, até atingir um erro menor. Sendo possível ter informações mais precisas