In [138…
```
#Linear Regression Model for Players
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [139…
```
#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 [140…
```
#Calculate Fantasy Points
df = df[['Player','Date','PTS','3P','TRB','AST','STL','BLK','TOV','MP']]

#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 [141…
```
#Sort dataframe for easier viewing of upcoming calculations
df = df.sort_values(['Player', 'Date'], ascending= [True, True], ignore_index=True)
```

In [142…
```
#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
#Create rolling average of last 3 games
df['Last3']= df.groupby('Player',sort=False).rolling(window=3, min_periods=1).FP.mean().
#create rolling average of last 5 games
df['Last5']= df.groupby('Player',sort=False).rolling(window=5, min_periods=1).FP.mean().
#creates rolling average of last 7 games
df['Last7']= df.groupby('Player',sort=False).rolling(window=7, min_periods=1).FP.mean().
#Cretes a rolling average. Window needs to be updated occasionally
df['Avg']= df.groupby('Player',sort=False).rolling(window=30, min_periods=1).FP.mean().r

df.head()
```

Out[142]:

	Player	Date	PTS	3P	TRB	AST	STL	BLK	TOV	MP	DD	TD	FP	Last3	Last5	Last7	Av
0	A.J. Green	2022-10-22	0	0	0	1	0	0	0	2	0	0	1.50	1.500000	1.500000	1.500000	1.50000
1	A.J. Green	2022-11-16	3	1	3	0	1	0	0	15	0	0	9.25	5.375000	5.375000	5.375000	5.37500
2	A.J. Green	2022-11-21	0	0	0	0	0	0	0	1	0	0	0.00	3.583333	3.583333	3.583333	3.58333
3	A.J. Green	2022-11-25	0	0	1	0	0	0	0	3	0	0	1.25	3.500000	3.000000	3.000000	3.00000
4	A.J. Green	2022-11-27	0	0	0	0	0	0	0	2	0	0	0.00	0.416667	2.400000	2.400000	2.40000

In [143…
```
df = df.drop(['Player','Date','DD','TD'],axis=1)
df.head()
```

Out[143]:

	PTS	3P	TRB	AST	STL	BLK	TOV	MP	FP	Last3	Last5	Last7	Avg

0	0	0	0	1	0	0	0	2	1.50	1.500000	1.500000	1.500000	1.500000
1	3	1	3	0	1	0	0	15	9.25	5.375000	5.375000	5.375000	5.375000
2	0	0	0	0	0	0	0	1	0.00	3.583333	3.583333	3.583333	3.583333
3	0	0	1	0	0	0	0	3	1.25	3.500000	3.000000	3.000000	3.000000
4	0	0	0	0	0	0	0	2	0.00	0.416667	2.400000	2.400000	2.400000

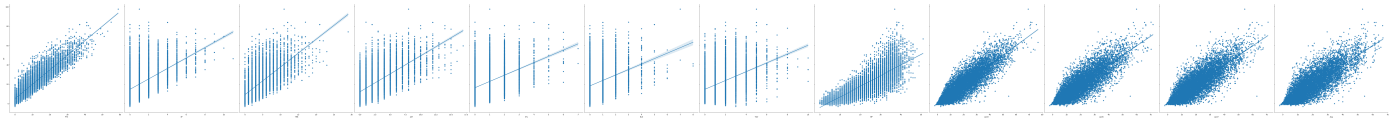
In [159…
```
df.describe()
```

Out[159]:

	PTS	3P	TRB	AST	STL	BLK	TOV	MP
count	9051.000000	9051.000000	9051.000000	9051.000000	9051.000000	9051.000000	9051.000000	9051.000000
mean	10.621257	1.145398	4.089493	2.346039	0.692741	0.459728	1.343940	22.732295
std	8.912823	1.464446	3.426731	2.573864	0.945855	0.817088	1.447891	10.853298
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	4.000000	0.000000	2.000000	0.000000	0.000000	0.000000	0.000000	15.000000
50%	9.000000	1.000000	3.000000	2.000000	0.000000	0.000000	1.000000	24.000000
75%	16.000000	2.000000	6.000000	3.000000	1.000000	1.000000	2.000000	32.000000
max	59.000000	11.000000	29.000000	17.000000	7.000000	8.000000	10.000000	51.000000

In [144…
```
#Using visualisation, you should be able to judge which variables have
#a linear relationship with y and each other. Start by using Seaborn's pairplot.
sns.pairplot(df,x_vars=['PTS','3P','TRB','AST','STL','BLK','TOV','MP','Last3','Last5','L
#sns.pairplot(df)
```

Out[144]:
```
<seaborn.axisgrid.PairGrid at 0x2120fdfd700>
```



In [145…
```
#It looks like our variables MP,PTS,Last3,Last5,Last7,Avg all have strong correlations.
#For now we will keep all the variables for our linear regression model.
#We will perform feature selection via trial and error when testing our linear regressio

#Also, not that MP seems to have a slightly non linear relationship with FP
```

In [146…
```
#Create a correlation matrix to show relationship between select variables
corr_matrix = df[['PTS','3P','TRB','AST','STL','BLK','TOV','MP','Last3','Last5','Last7',
corr_matrix
```

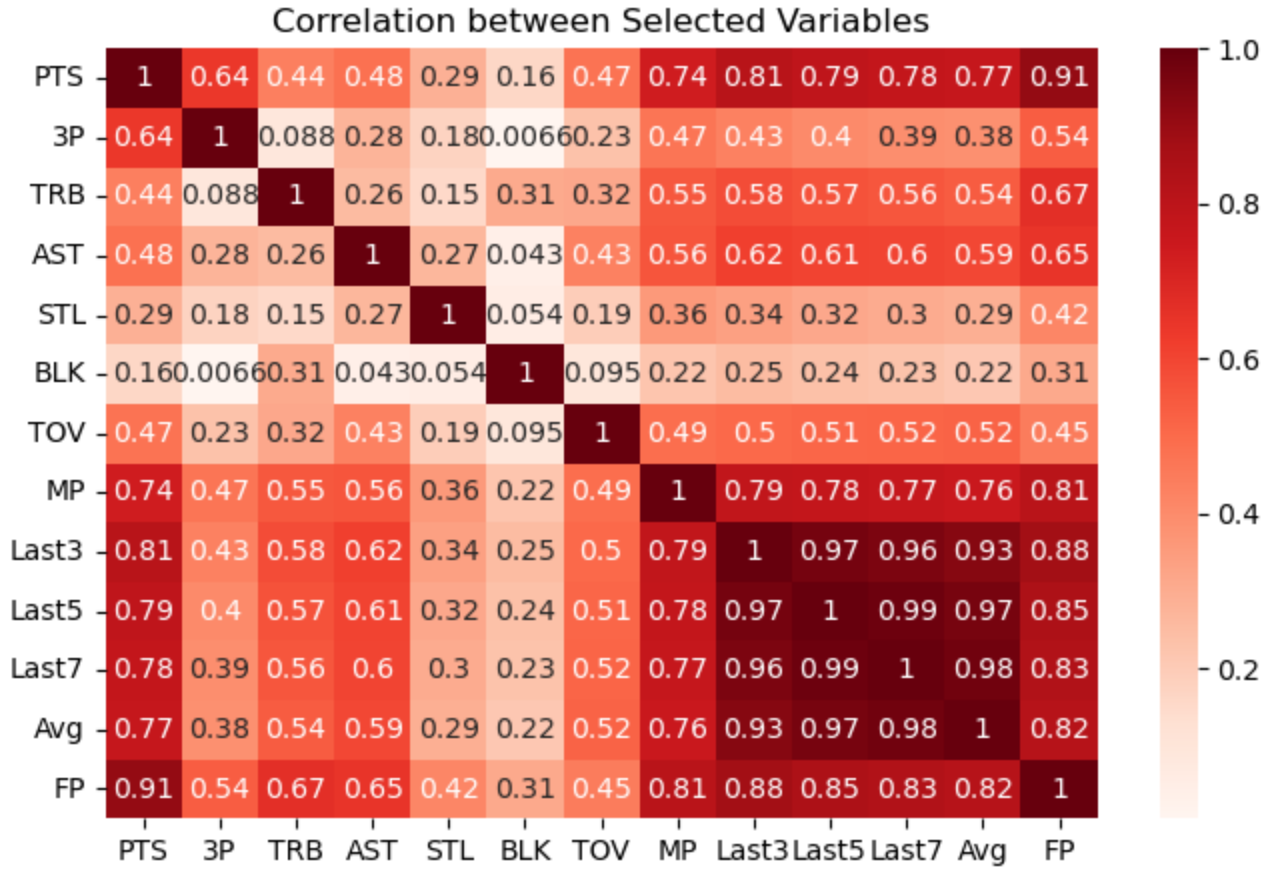
Out[146]:

	PTS	3P	TRB	AST	STL	BLK	TOV	MP	Last3	Last5	Last7
PTS	1.000000	0.641289	0.437349	0.483661	0.288539	0.163623	0.474910	0.739367	0.810536	0.787553	0.778396
3P	0.641289	1.000000	0.088411	0.277574	0.175448	0.006556	0.234422	0.470430	0.431587	0.403714	0.394012
TRB	0.437349	0.088411	1.000000	0.257524	0.154908	0.305793	0.316054	0.551622	0.584455	0.565644	0.555706
AST	0.483661	0.277574	0.257524	1.000000	0.269393	0.042933	0.431614	0.559096	0.616075	0.606469	0.601546
STL	0.288539	0.175448	0.154908	0.269393	1.000000	0.053830	0.193603	0.358795	0.339627	0.315927	0.304547
BLK	0.163623	0.006556	0.305793	0.042933	0.053830	1.000000	0.095349	0.220792	0.251159	0.235688	0.227779
TOV	0.474910	0.234422	0.316054	0.431614	0.193603	0.095349	1.000000	0.488762	0.503816	0.513463	0.517051

MP	0.739367	0.470430	0.551622	0.559096	0.358795	0.220792	0.488762	1.000000	0.786483	0.775043	0.76881
Last3	0.810536	0.431587	0.584455	0.616075	0.339627	0.251159	0.503816	0.786483	1.000000	0.970302	0.95536
Last5	0.787553	0.403714	0.565644	0.606469	0.315927	0.235688	0.513463	0.775043	0.970302	1.000000	0.98766
Last7	0.778396	0.394012	0.555706	0.601546	0.304547	0.227779	0.517051	0.768810	0.955368	0.987660	1.00000
Avg	0.768007	0.383247	0.542021	0.593651	0.292721	0.215961	0.521841	0.757249	0.934691	0.966240	0.98022
FP	0.909010	0.537849	0.666789	0.650862	0.416565	0.313074	0.447867	0.808922	0.877412	0.848062	0.83496

In [147…

```
#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()
```



In [148…

```
from statsmodels.formula.api import ols
```

In [149…

```
#After some trial and error I settled on Last 3 and Last 5
FFPG_vs_features = ols('FP ~ Last3+Last5',data=df)
FFPG_vs_features = FFPG_vs_features.fit()
print(FFPG_vs_features.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                     FP   R-squared:                       0.770
Model:                            OLS   Adj. R-squared:                  0.770
Method:                 Least Squares   F-statistic:                 1.515e+04
Date:                Fri, 16 Dec 2022   Prob (F-statistic):               0.00
Time:                        11:14:23   Log-Likelihood:                -30535.
No. Observations:                9051   AIC:                         6.108e+04
Df Residuals:                    9048   BIC:                         6.110e+04
Df Model:                           2
Covariance Type:            nonrobust
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.2468      0.143      1.726      0.084      -0.033       0.527
Last3          1.0582      0.024     44.720      0.000       1.012       1.105
Last5         -0.0657      0.024     -2.700      0.007      -0.113      -0.018
==============================================================================
Omnibus:                      336.228   Durbin-Watson:                   2.227
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              760.832
Skew:                           0.223   Prob(JB):                    6.13e-166
Kurtosis:                       4.349   Cond. No.                         66.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

In [150…
```
#Set up for splitting our data into test and train, and then see testing our linear regr
```

In [151…
```
#Set up our features variables
feature_cols = ['Last3','Last5']
X=df[feature_cols]
```

In [152…
```
from statsmodels.formula.api import ols
```

In [153…
```
#Set up our target variable
y=df['FP']
```

In [154…
```
#Splitting X & y into training and testing sets:
#By passing our X and y variables into the train_test_split method,
#we are able to capture the splits in data by assigning 4 variables to the result.

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y)
```

In [155…
```
#Train Model
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train,y_train)
```

Out[155]:
```
LinearRegression()
```

In [156…
```
print(linreg.intercept_)
print(linreg.coef_)
zip(feature_cols,linreg.coef_)
```

Out[156]:
```
0.2066319863881212
[ 1.06346886 -0.06840659]
<zip at 0x2126f306640>
```

In [157…
```
#Make predictions based upon our model
y_pred = linreg.predict(X_test)
```

In [158…
```
#Model Evaluation via Mean Absolute error, Mean Squared Error
#and Root Mean Squared error
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,y_pred))
print(metrics.mean_squared_error(y_test, y_pred))
print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
5.328164589714438
50.6881264292958
```

7.119559426628575