Module 8: Project Code and Data Analysis

Lionel Kevin Mbuyi Mukanya

Colorado State University Global

MIS581: Capstone: Business Intelligence and Data Analytics

Chung, Steve

11/5/2023

https://github.com/lionelmukanya/Capstone-Project

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from matplotlib import pyplot
from xgboost import plot_importance
import seaborn as snsb
from sklearn.ensemble import RandomForestRegressor
```

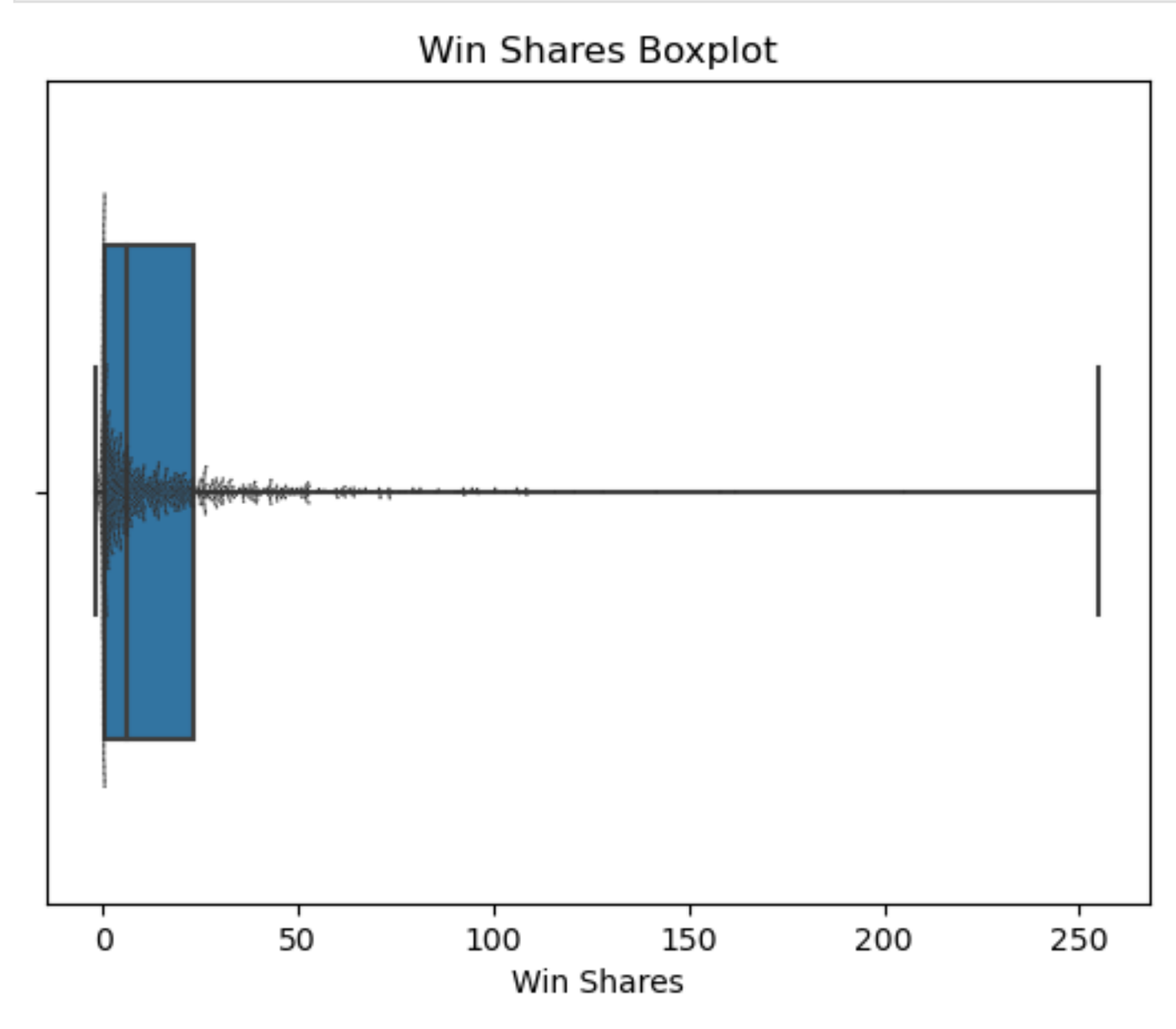
In [2]:
```python
df = pd.read_excel('NBA Draft.xlsx')
```

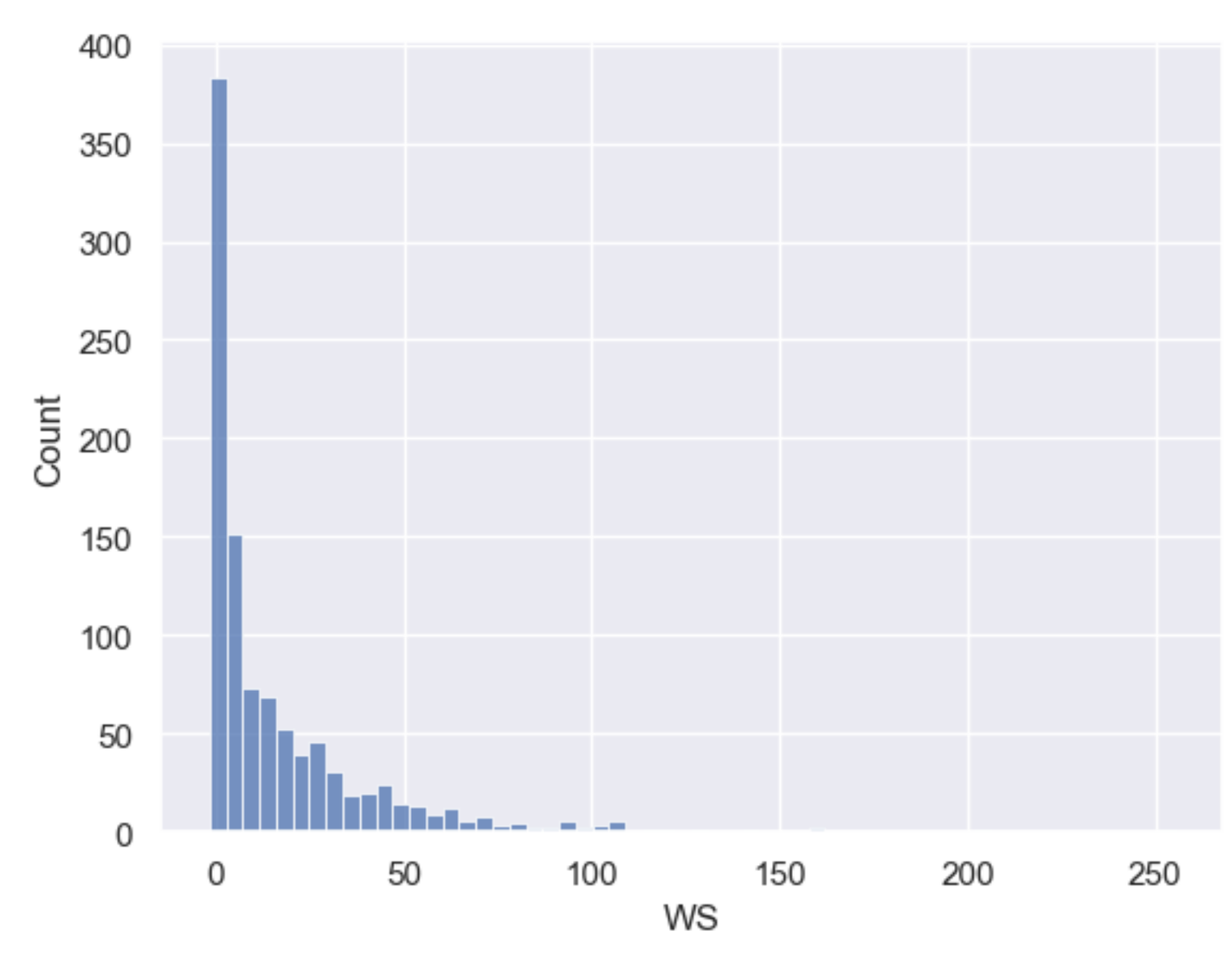
In [3]:
```python
df[['WS','PPG', 'RBPG','ASTPG','3P%', 'FT%']].describe()
```

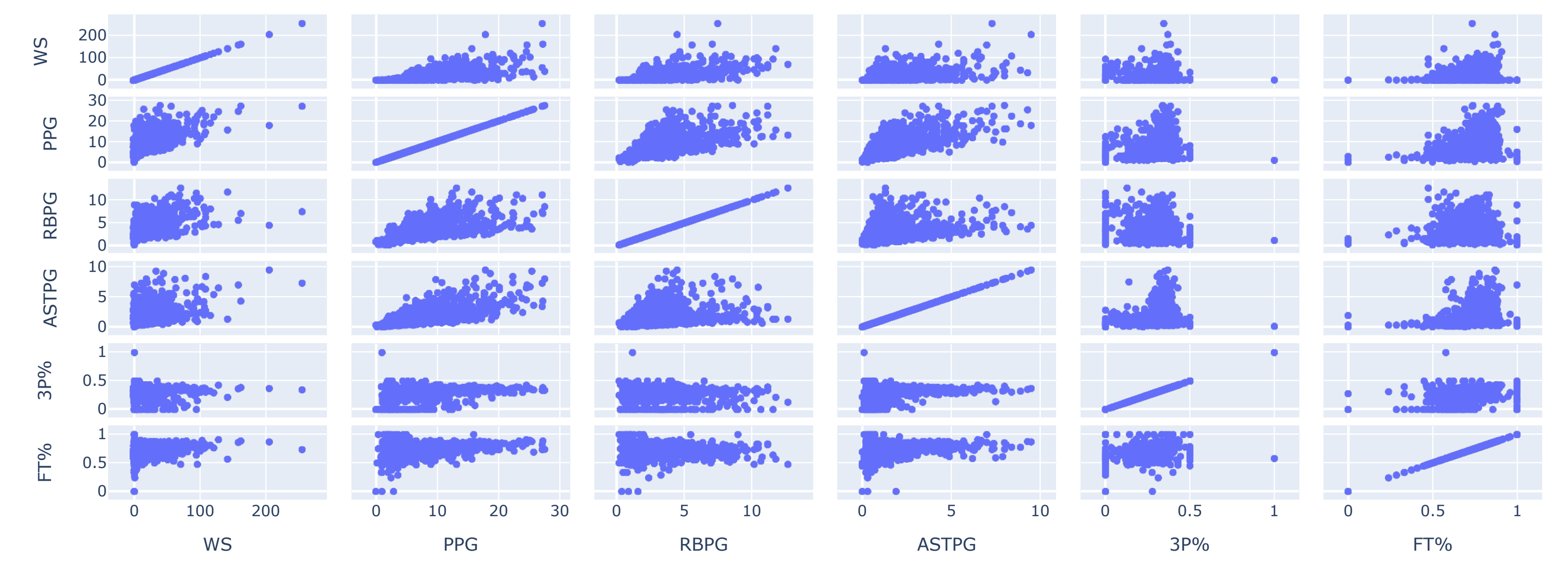
Out[3]:

	WS	PPG	RBPG	ASTPG	3P%	FT%
count	1007.000000	1007.000000	1007.000000	1007.000000	1007.000000	1007.000000
mean	16.618669	7.881132	3.336147	1.681033	0.290622	0.733435
std	25.252120	5.087218	2.095499	1.512600	0.113478	0.114308
min	-1.700000	0.000000	0.200000	0.000000	0.000000	0.000000
25%	0.600000	4.000000	1.900000	0.700000	0.265000	0.674000
50%	6.100000	6.800000	2.900000	1.200000	0.325000	0.750000
75%	23.000000	10.450000	4.300000	2.100000	0.360000	0.808000
max	255.100000	27.600000	12.700000	9.500000	1.000000	1.000000

In [4]:
```python
#Win Shares Boxplot
#Create horizontal boxplot with points
snsb.boxplot(x=df['WS'], data= df['WS'], whis=[0, 100], width=.6, fliersize=5, orient="h")
snsb.swarmplot(x=df['WS'], data= df['WS'], color=".2", size=1, orient="h")

#Add title and axis labels
plt.title("Win Shares Boxplot")
plt.xlabel("Win Shares")
plt.show()


snsb.set(style="darkgrid")

snsb

snsb.histplot(data=df['WS'], x=df['WS'])
plt.show()


import plotly.express as px
fig = px.scatter_matrix(df[['WS','PPG', 'RBPG','ASTPG','3P%', 'FT%']])
fig.show()
```







In [5]:
```python
#Statistical Details of the dataset
print(df.head())
```

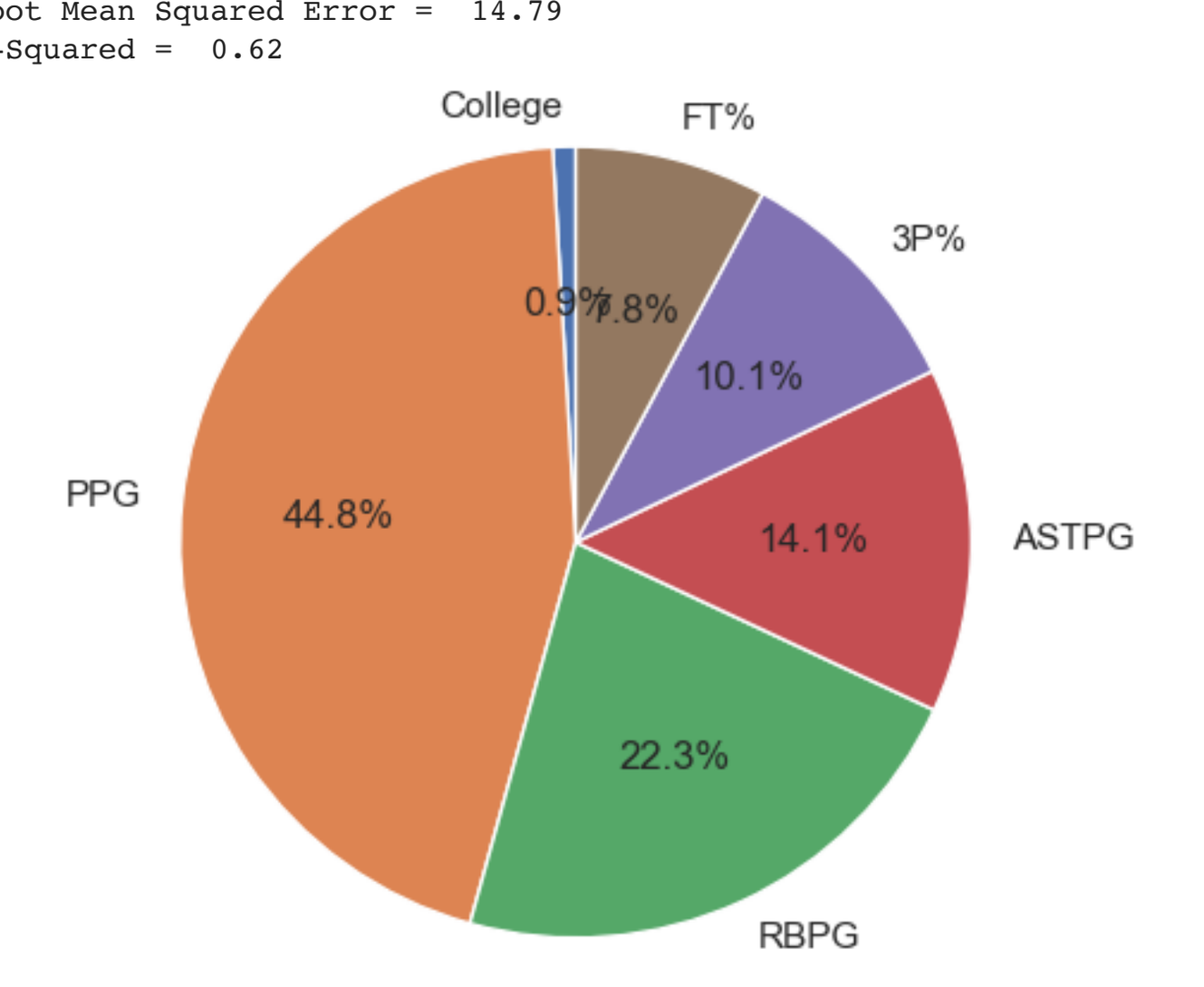
```
   Rk  Pk   Tm                Player  Year Drafted  College  Yrs    G     MP  \
0   1   1  HOU              Yao Ming          2002        0    8  486  15818
1   2   2  CHI          Jay Williams          2002        1    1   75   1961
2   3   3  GSW         Mike Dunleavy          2002        1   15  986  27340
3   4   4  MEM           Drew Gooden          2002        1   14  790  20127
4   5   5  DEN  Nikoloz Tskitishvili          2002        0    4  172   1946

     PTS  ...    3P%    FT%   MPG   PPG  RBPG  ASTPG    WS   WS/48  BPM  VORP
0   9247  ...  0.200  0.833  32.5  19.0   9.2    1.6  65.9   0.200  3.2  20.7
1    714  ...  0.322  0.640  26.1   9.5   2.6    4.7   0.8   0.020 -2.0   0.0
2  11048  ...  0.377  0.803  27.7  11.2   4.3    2.2  58.5   0.103  0.3  15.7
3   8653  ...  0.257  0.760  25.5  11.0   7.1    1.1  43.9   0.105 -1.2   4.3
4    507  ...  0.235  0.730  11.3   2.9   1.8    0.7  -1.6  -0.039 -6.1  -2.0

[5 rows x 23 columns]
```

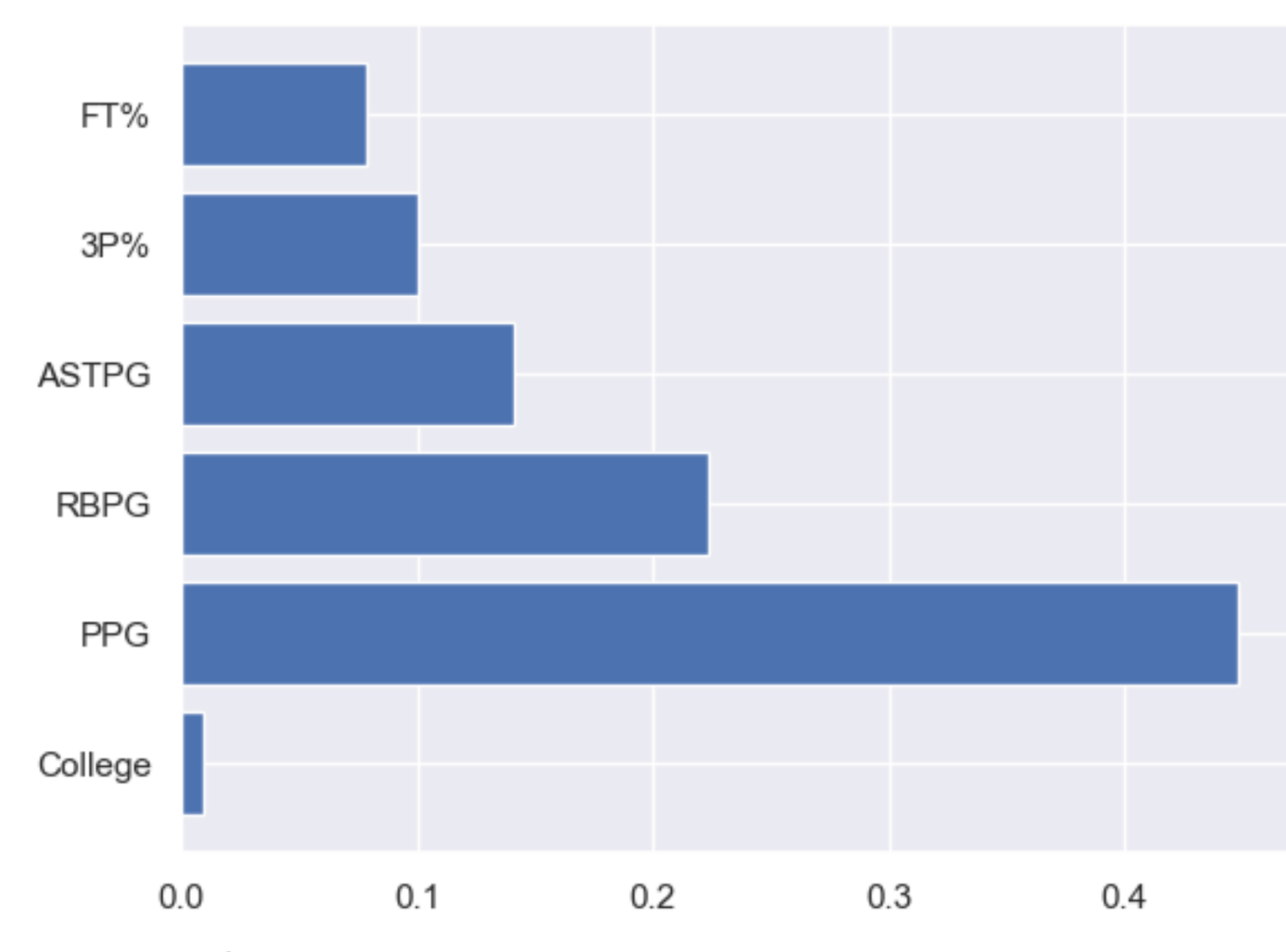
In [6]:
```python
#Linear Regression
from sklearn import linear_model
import statsmodels.api as sm
X = df[['College','PPG', 'RBPG','ASTPG','3P%', 'FT%']]
y = df['WS']


x = sm.add_constant(X) # adding a constant

model = sm.OLS(y, x).fit()
predictions = model.predict(x)

print_model = model.summary()
print(print_model)

mse_linear_regression = round(mean_squared_error(y, predictions), 2)
rmse_linear_regression = round(mse_linear_regression**.5, 2)
R2_linear = round(r2_score(y, predictions), 2)
print("Mean Squared Error = ",mse_linear_regression)
print("Root Mean Squared Error = ", rmse_linear_regression)
print("R-Squared = ",R2_linear)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                     WS   R-squared:                       0.514
Model:                            OLS   Adj. R-squared:                  0.511
Method:                 Least Squares   F-statistic:                     176.2
Date:                Sun, 05 Nov 2023   Prob (F-statistic):          7.77e-153
Time:                        23:46:50   Log-Likelihood:                -4316.7
No. Observations:                1007   AIC:                             8647.
Df Residuals:                    1000   BIC:                             8682.
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -14.4833      4.263     -3.398      0.001     -22.849      -6.118
College       -2.3625      1.476     -1.600      0.110      -5.260       0.535
PPG            1.7927      0.228      7.875      0.000       1.346       2.239
RBPG           3.9199      0.406      9.651      0.000       3.123       4.717
ASTPG          2.4590      0.547      4.495      0.000       1.386       3.533
3P%           -6.5026      5.774     -1.126      0.260     -17.833       4.828
FT%            4.9080      5.710      0.859      0.390      -6.298      16.114
==============================================================================
Omnibus:                      559.064   Durbin-Watson:                   1.419
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            11906.456
Skew:                           2.085   Prob(JB):                         0.00
Kurtosis:                      19.321   Cond. No.                         130.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Mean Squared Error =  309.67
Root Mean Squared Error =  17.6
R-Squared =  0.51
```

In [8]:
```python
#Random Forest

X = df[['College','PPG', 'RBPG','ASTPG','3P%', 'FT%']]

y = df['WS']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=44)
from sklearn.ensemble import RandomForestRegressor
rf_model = RandomForestRegressor(n_estimators=50, max_features="auto", random_state=44)
rf_model.fit(X_train, y_train)
predictions = rf_model.predict(X_test)
importances = rf_model.feature_importances_
columns = X.columns
i=0

while i < len(columns):
    print(f" The importance of feature '{columns[i]}' is {round(importances[i] * 100, 2)}%.")
    i += 1
mse_random_forest = round(mean_squared_error(y_test, predictions),2)
rmse_random_forest = round(mse_random_forest**.5, 2)
R2_random_forest = round(r2_score(y_test, predictions), 2)
print("Mean Squared Error = ",mse_random_forest)
print("Root Mean Squared Error = ", rmse_random_forest)
print("R-Squared = ",R2_random_forest)

labels = "College", "PPG", "RBPG", "ASTPG", "3P%", "FT%"
fig, ax = plt.subplots()
ax.pie(importances,  labels=labels, autopct='%1.1f%%',
        shadow=False, startangle=90)
ax.axis('equal')
plt.show()


plt.barh(columns, importances)
plt.show()
print("Mean Squared Error = ",mse_random_forest)
print("Root Mean Squared Error = ", rmse_random_forest)
print("R-Squared = ",R2_random_forest)
```

```
 The importance of feature 'College' is 0.93%.
 The importance of feature 'PPG' is 44.76%.
 The importance of feature 'RBPG' is 22.34%.
 The importance of feature 'ASTPG' is 14.09%.
 The importance of feature '3P%' is 10.05%.
 The importance of feature 'FT%' is 7.83%.
Mean Squared Error =  218.73
Root Mean Squared Error =  14.79
R-Squared =  0.62
```

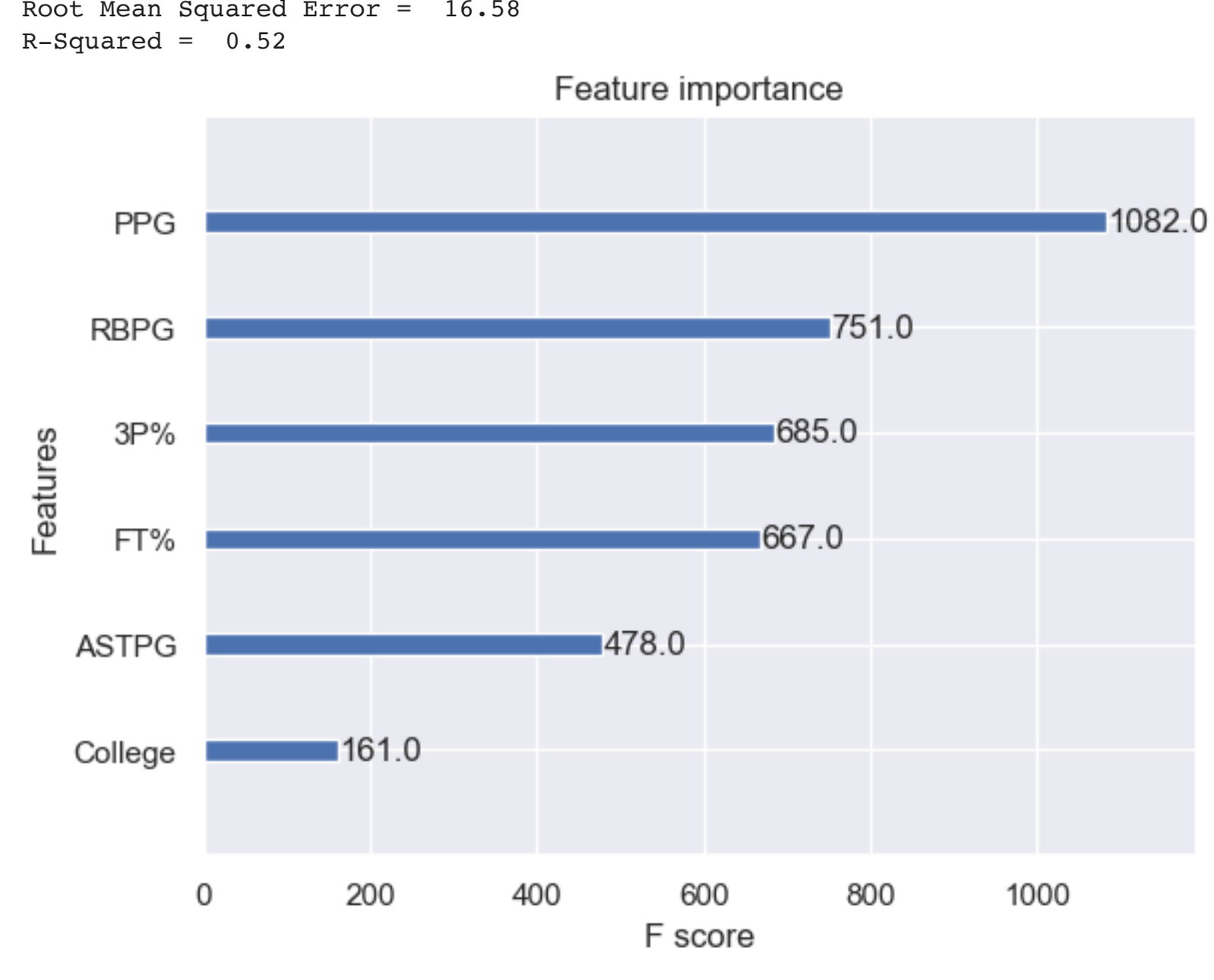





```
Mean Squared Error =  218.73
Root Mean Squared Error =  14.79
R-Squared =  0.62
```

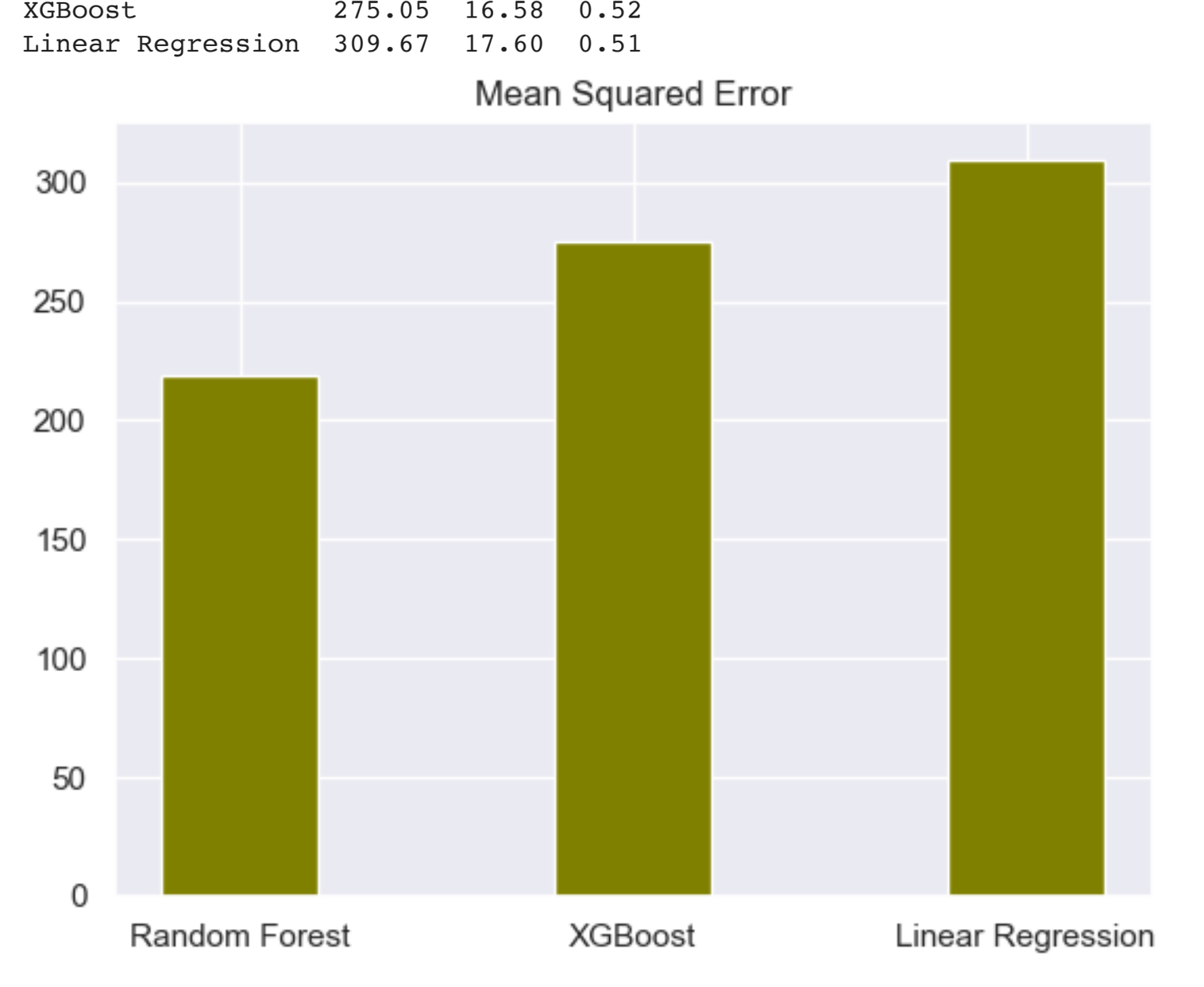
In [9]:
```python
# XGBboost
from sklearn.inspection import permutation_importance
import xgboost as xgb
X = df[['College','PPG','RBPG','ASTPG','3P%', 'FT%']]
y = df['WS']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=44)

#Creating an XGBoost regressor
xgbmodel = xgb.XGBRegressor()

#Training the model on the training data
xgbmodel.fit(X_train, y_train)

#Making predictions on the test set
predictions = xgbmodel.predict(X_test)
importances = xgbmodel.feature_importances_

mse_xgb = round(mean_squared_error(y_test, predictions),2)
rmse_xgb = round(mse_xgb**.5, 2)
R2_xgb = round(r2_score(y_test, predictions), 2)
print("Mean Squared Error = ",mse_xgb)
print("Root Mean Squared Error = ", rmse_xgb)
print("R-Squared = ",R2_xgb)

plot_importance(xgbmodel)
pyplot.show()
```

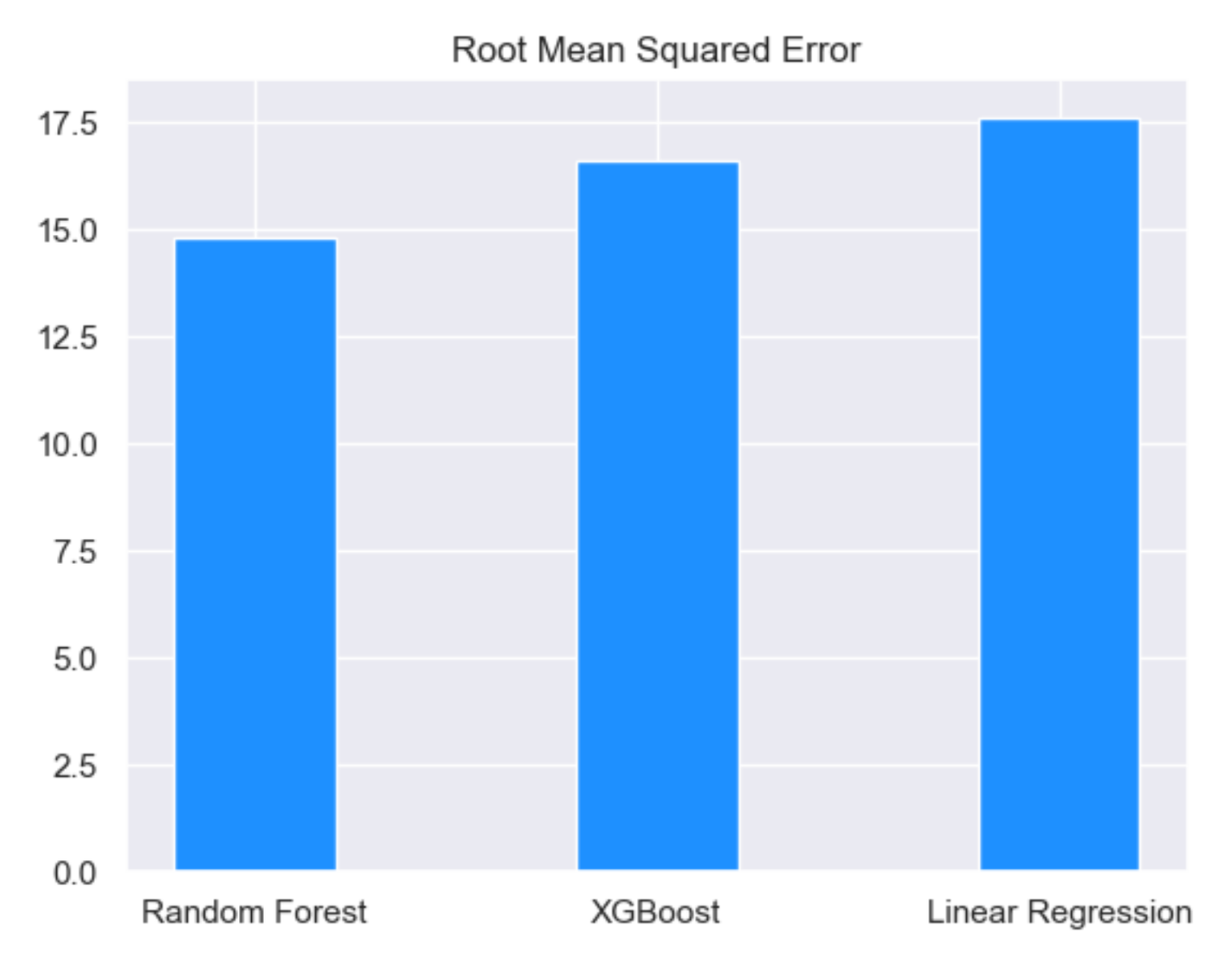
```
Mean Squared Error =  275.05
Root Mean Squared Error =  16.58
R-Squared =  0.52
```



In [10]:
```python
data = {'Random Forest':[mse_random_forest, rmse_random_forest , R2_random_forest],
        'XGBoost': [round(mse_xgb, 2), round(rmse_xgb, 2), round(R2_xgb, 2)],
       'Linear Regression': [mse_linear_regression, rmse_linear_regression, R2_linear]}
a = pd.DataFrame.from_dict(data, orient='index', columns= ["MSE", "RMSE", "R²"])\

print(a)

# creating the bar plots
plt.bar(["Random Forest", "XGBoost", "Linear Regression"],a['MSE'], color ='olive',
        width = 0.4)
plt.title("Mean Squared Error")
plt.show()

plt.bar(["Random Forest", "XGBoost", "Linear Regression"],a['RMSE'], color ='dodgerblue',
        width = 0.4)
plt.title("Root Mean Squared Error")
plt.show()

plt.bar(["Random Forest", "XGBoost", "Linear Regression"],a['R²'], color ='tomato',
        width = 0.4)
plt.title("R-Squared")
plt.show()
```

```
                      MSE   RMSE    R²
Random Forest      218.73  14.79  0.62
XGBoost            275.05  16.58  0.52
Linear Regression  309.67  17.60  0.51
```





In [ ]: