Prop Bet - LeBron James

Ken Marciel

7/22/2021

Packages

```
# Packages installed in a terminal session:
## pip install --user matplotlib
## pip install --user numpy
## pip install --user pandas
## pip install --user scipy
## pip install --user seaborn
## pip install --user sklearn
## pip install --user statsmodels

# Packages loaded
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.stats as stats
import seaborn as sns
from sklearn import feature_selection
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.statespace.sarimax import SARIMAX

```

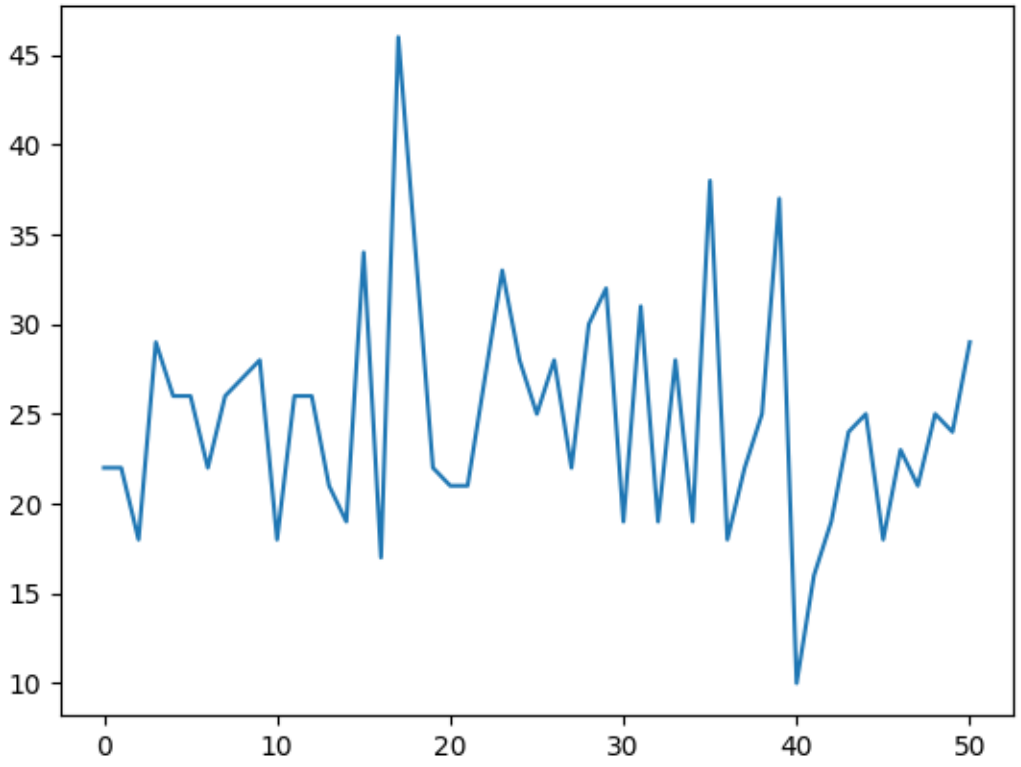
1. Data Collection

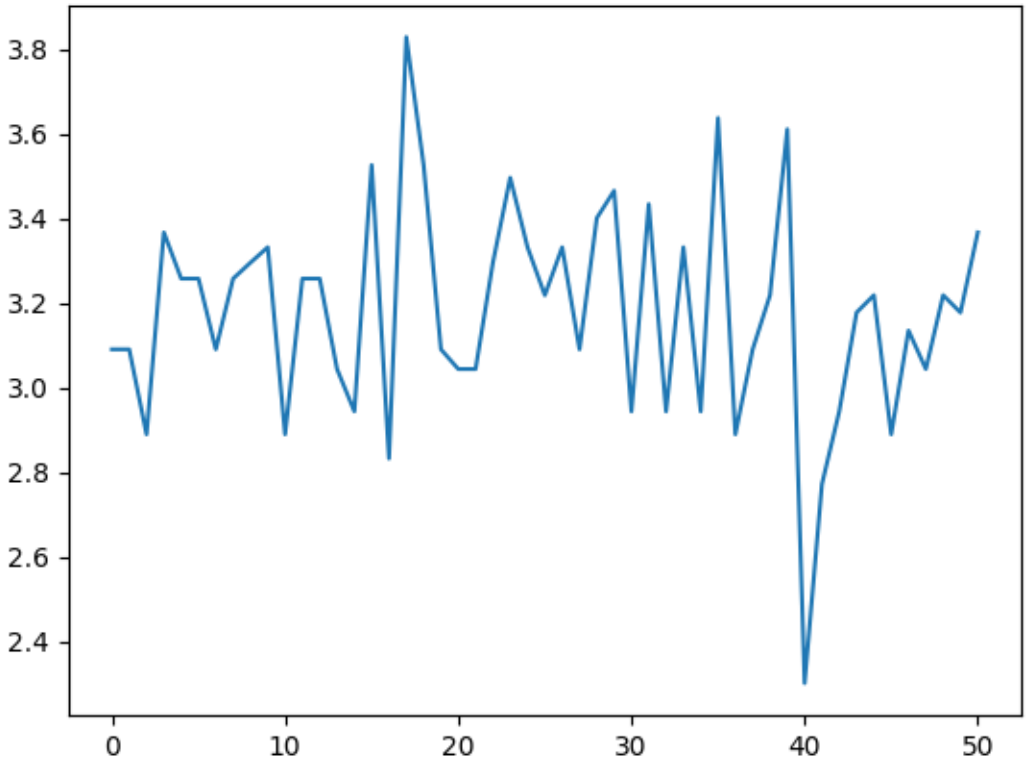
Data was collected from the website, Basketball-Reference.com, by Sports Reference. The data set consists of all the games played by LeBron James in the most recent season. This includes the 2020-2021 regular season and the six games played by in the 2021 playoffs, downloaded as two Excel files, respectively. Using Excel, the files were converted to CSV files. These were then read into Python. The subsets of all games that LeBron James played in were concatenated to exclude the games in which he was inactive. The index of the new file was renumbered from 0 to 50, for the 51 games that LeBron James played in for 2020-2021 regular season and playoffs.

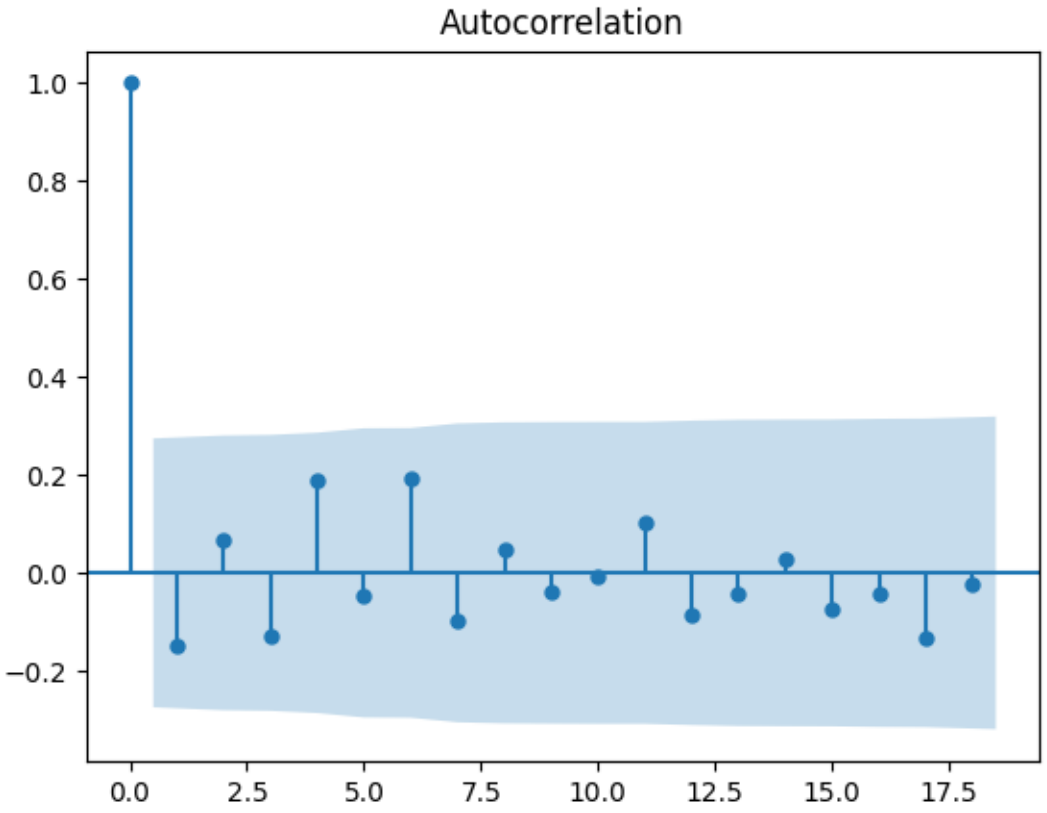
```
# LeBron James: 2020-21 Regular Season
# Source: https://www.basketball-reference.com/players/j/jamesle01/gamelog/2021
data1 = 'C:/Users/keoka/Desktop/python_work/lbj_2020_21_regular_season.csv'
df1 = pd.read_csv(data1)
df1.head()
len(df1) # 72 regular season games

# LeBron James: 2021 Playoffs
# Source: https://www.basketball-reference.com/players/j/jamesle01/gamelog/2021
data2 = 'C:/Users/keoka/Desktop/python_work/lbj_2021_playoffs.csv'
df2 = pd.read_csv(data2)
df2.head()
len(df2) # 6 playoff games

# LeBron James: 2020-21 Combined Regular Season and Playoffs
df3 = pd.concat([df1, df2])
df3.head()
len(df3)  # 78 total games for 2020-21

# Remove missing values (for inactivity) from data series
a = df3[0:35 + 1]  # first 36 games played
b = df3[37:41 + 1]  # next 5 games played
c = df3[62:63 + 1]  # next 2 games played
d = df3[70:77 + 1]  # last 8 games played
df4 = pd.concat([a,b,c,d])
df4 = df4.reset_index()  # renumber from 0 to 50
len(df4)  # played in 51 total games for 2020-21

```

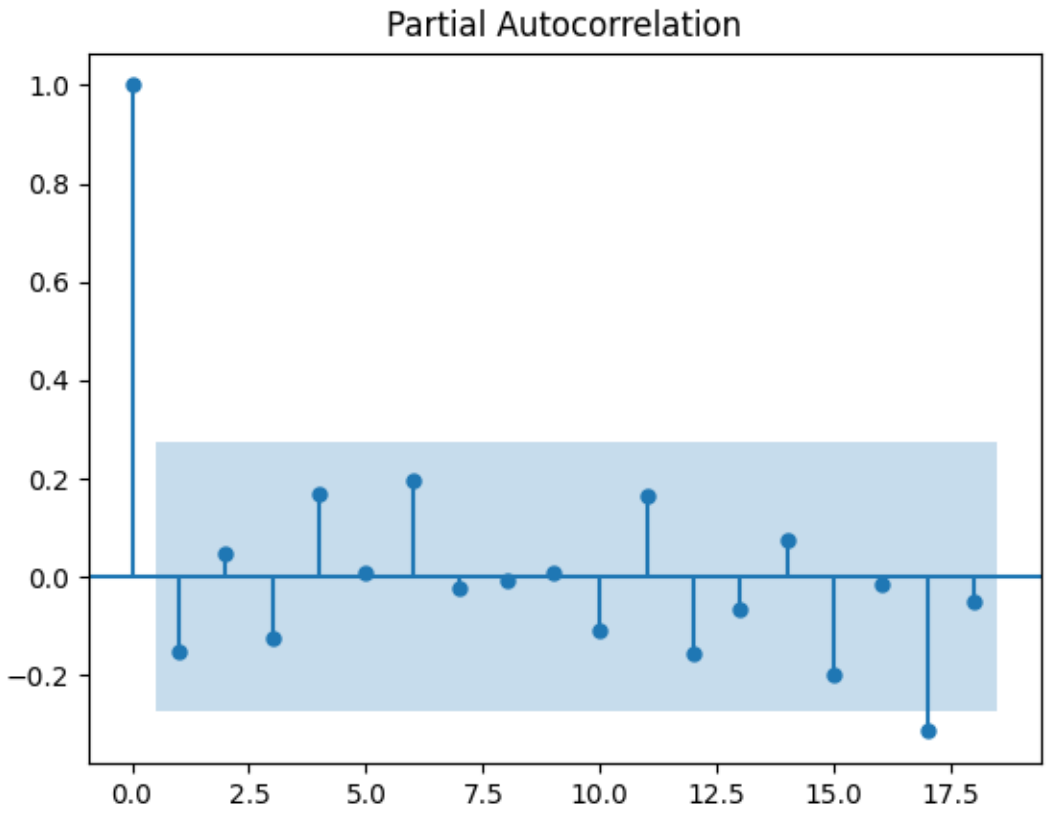
2. Data Transformation

A plot of the time series data displays no noticeable trend, so the series appears stationary. The plot suggests the presence of heteroscedasticity (time-varying variance) and possible autocorrelation. After applying a log transformation to the data, the plot shows stabilized variance. The correlograms (ACF and PACF plots) show a lack of evidence for significant autocorrelation (within the confidence band of $\pm 2/\sqrt{(n)}$).


45
40
35
30
25
20
15
10
0
10
20
30
40
50


3.8
3.6
3.4
3.2
3.0
2.8
2.6
2.4
0
10
20
30
40
50

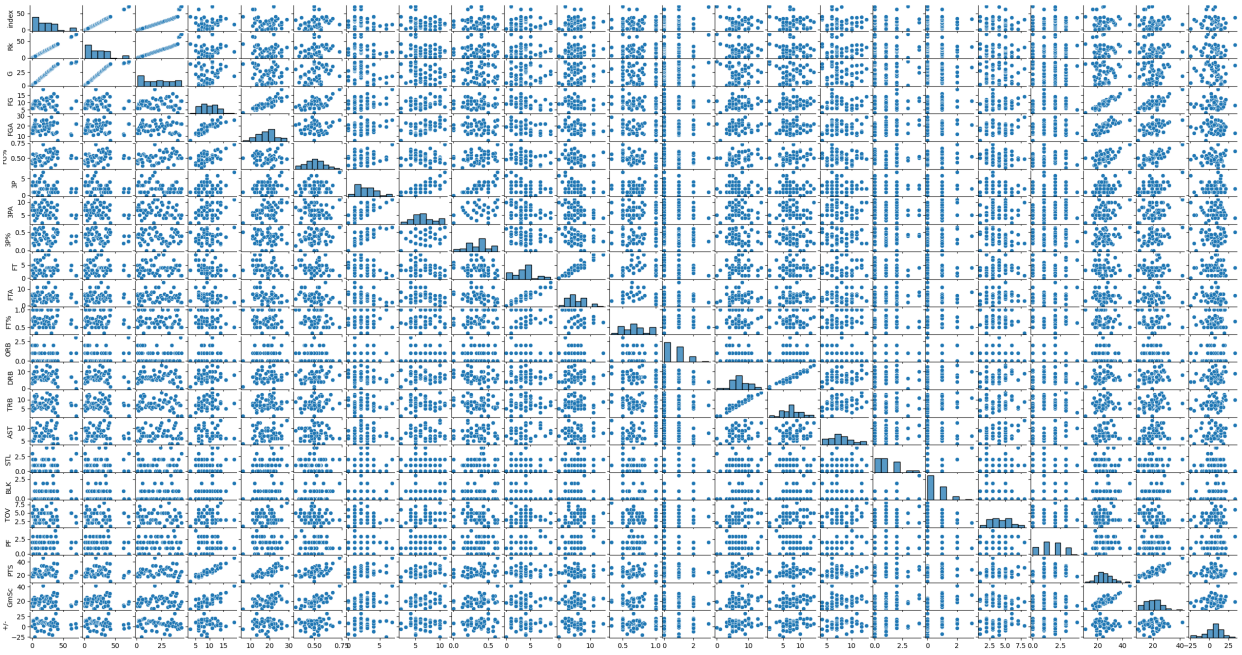

Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5




```
# Plot the time series data
plt.plot(df4['PTS'])
plt.show()
## There is no noticeable trend, so the series appears
## stationary. The plot suggests the presence of
## heteroscedasticity and possible autocorrelation.

# Log transformations to stabilize variance
df5 = np.log(df4['PTS'])

# Plots of transformed data
plt.plot(df5)  # heteroscedasticity has been reduced
plot_acf(df5)  # lack of evidence for autocorrelation
plot_pacf(df5)  # lack of evidence for autocorrelation
plt.show()

```

3. Model Features

Of the 30 columns in the data file, the 22 metrics recorded for each game were the quantitative features under consideration to formulate a predictive model. A scatterplot matrix reveals that Points (PTS) exhibit the strongest correlation with Field Goals (FG) and Game Score (GmSc), respectively.



```
# Scatterplot matrix
sns.pairplot(df4)
plt.show()
## PTS exhibits strongest correlation with FG and GmSc
df4['PTS'].corr(df4['FG'])  # 0.9387
df4['PTS'].corr(df4['GmSc'])  # 0.8608
df4['FG'].corr(df4['GmSc'])  # 0.7739

```

4. Modeling Process

Four models were fit parsimoniously. The first model regresses PTS onto FG. The second model regresses PTS onto GmSc. The third model regresses PTS onto FG and GmSc. The fourth model adds an FG*GmSc interaction term to the third model.

```
# Linear model 1: Field Goals
x = df4[['FG']]
y = df4['PTS']
reg1 = linear_model.LinearRegression()
reg1.fit(x,y)

# Linear model 2: Game Score
x = df4[['GmSc']]
y = df4['PTS']
reg2 = linear_model.LinearRegression()
reg2.fit(x,y)

# Linear model 3: Field Goals + Game Score
x = df4[['FG', 'GmSc']]
y = df4['PTS']
reg3 = linear_model.LinearRegression()
reg3.fit(x,y)

# Linear model 4: FG + GmSc + FG*GmSc
x = df4[['FG', 'GmSc']]
interaction = PolynomialFeatures(degree=1)
x = interaction.fit_transform(x)
y = df4['PTS']
reg4 = linear_model.LinearRegression()
reg4.fit(x,y)

```

5. Model Testing and Evaluation

The third regression model was selected because it had the highest coefficient of determination (R^2) of 0.9262. This suggests that about 93 percent of the variation in the data is explained by the model. The fourth regression model had the same coefficient of determination, suggesting that the interaction between Field Goals and Game Score is not significant. A one-step-ahead forecast was employed, using the values of FG and and GmSc from the last playoff game of 2021, to predict that LeBron James will score 28 points in the second game of the next preseason, on October 6, 2021, between the L.A. Lakers and the Phoenix Suns, in Phoenix.

```
# Coefficient of determination
reg1.score(x,y)  # 0.8812
reg2.score(x,y)  # 0.7410
reg3.score(x,y)  # 0.9262
reg4.score(x,y)  # 0.9262

# Selected model
intercept = reg3.intercept_  # 4.955
coef = reg3.coef_  # [1.395, 0.3394]

# Prediction using data from final game of 2021 playoffs
intercept + coef[0]*11 + coef[1]*22.8  # 28 points

# Same prediction using different code
dim1 = np.array([[11],[22.8]])
dim2 = np.reshape(dim1, (1,1))
reg3.predict(dim2)
```