BMI and the NBA: A Statistical and Machine Learning Analysis

The body mass index (BMI) is a point score value that is calculated based on someone's height and weight. Although BMI is not the be-all and end-all metric to evaluate someone's body composition and performance, it is widely agreed upon to be a useful metric. A major factor that skews the BMI metric is an individual's fat free mass index (FFMI). FFMI is the measure of fat free mass, which is composed of the weight of bones, organs, muscles, etc. The FFMI is also a point value score that is calculated based on someone's height, weight and body fat percentage. Generally speaking, the higher weight and the lower the body fat percentage, the higher the FFMI score. It is generally accepted that an FFMI score of 25 is the limit of what can be attained naturally (ie. without the use of steroids).

1. The NBA Bios dataset

For this write up, we will be using the NBA bios taken from:

https://stats.nba.com/players/bio/

Ensuring we have the correct working directory set, we will import the dataset:

```
import pandas as pd
import numpy as np

playerbios2020 = pd.read_csv("G:/Dropbox/Software Development/FILES/SATB/NBA/2020 NBA Player
Bios.csv")

playerbios2020.head()
```

	Player	Team	Age	Height	Weight	College	Country	Draft Year	Draft Round	Draft Number	GP	PTS	REB	AST	NetRtg	OREB%	DREB%	USG%
0	Aaron Gordon	ORL	24	6-8	235	Arizona	USA	2014	1	4	26	13.2	7.0	2.9	-3.5	5.20%	16.10%	19.90%
1	Aaron Holiday	IND	23	6-0	185	UCLA	USA	2018	1	23	28	9.9	2.6	2.7	2.8	1.70%	8.90%	19.60%
2	Abdel Nader	OKC	26	6-5	225	Iowa State	Egypt	2016	2	58	22	7.4	1.8	0.7	-1.0	2.00%	7.40%	15.90%
3	Admiral Schofield	WAS	22	-	-	Tennessee	United Kingdo...	2019	2	42	15	3.9	1.3	0.7	6.1	1.60%	8.80%	12.40%
4	Al Horford	PHI	33	6-9	240	Florida	Dominican Rep...	2007	1	3	27	13.3	6.7	3.8	7.6	5.80%	15.70%	18.90%

One player of interest that was missing from this data was Zion Williamson. Zion was selected as the first draft pick in the 2019 NBA Draft and is predicted to make waves in the league. Taking his stats from:

https://en.wikipedia.org/wiki/Zion_Williamson

We add a row entry for Zion in our playerbios2020 dataframe using the following code:

```
ziondata = [{'Player' : 'Zion Williamson',
                         'Team' : 'NOP',
                         'Age' : 19.0,
                         'Height' : '6-6',
                         'Weight' : '284',
                         'College' : 'Duke',
                         'Country' : 'USA',
                         'Draft Year' : '2019',
                         'Draft Round' : '1',
                         'Draft Number' : '1',
                         'GP' : np.nan,
                         'PTS' : np.nan,
                         'REB' : np.nan,
                         'AST' : np.nan,
                         'NetRtg' : np.nan,
                         'OREB%' : np.nan,
                         'DREB%' : np.nan,
                         'USG%' : np.nan,
                         'TS%' : np.nan,
                         'AST%' : np.nan}]

ziondf = pd.DataFrame(ziondata)
playerbios2020 = playerbios2020.append(ziondf)
```

2. The BMI Calculation

Now that we have added Zion Williamson in our dataframe, a few other details need to be taken care of. Firstly, not every row entry in playerbios2020 has valid entries for height and weight. Null entries currently have a placeholder of '-'. Let's sort our dataframe by weight, reset the numerical index and replace these entries with NaN values. We can then count the number of NaN entries in the 'Height' and 'Weight' columns in preparation for our BMI calculation.

```
# Sort values by weight
playerbios2020.sort_values('Weight', ascending = False, inplace = True)
playerbios2020.reset_index(drop = True, inplace = True)

# Some height and weight values are missing and have '-' as a placeholder, let's replace with NaN
playerbios2020.replace('-', np.nan)

# Let's count how many NaN entries there are in Height and Weight columns
# Running this line returns 50 values
playerbios2020.isna().sum()

# Let's create a new dataframe without the NaN height and weight entries instead of dropping them
# from our original dataframe
bmidata = playerbios2020.iloc[:len(playerbios2020) - 50]
bmidata['Weight'] = bmidata['Weight'].astype(float)
```

Notice that height is currently entered in a string format of 'feet-inches'. We can use some string manipulation techniques to return a numerical value. We will aim to return the height values in inch measurements.

```
lis = bmidata['Height']

heightlist = []
for i in lis:
    feetinch = i.split('-')
    feet = feetinch[0]
    inch = feetinch[1]
    feet = float(feet)
    inch = float(inch)
    heightinch = (feet * 12) + inch
    heightlist.append(heightinch)

# heightlist is a list of all heights in inches in the same ordering as the bmidata dataframe
# Let's convert heightlist into a series which we can insert into bmidata as a column
heightseries = pd.DataFrame(heightlist)
heightseries.rename(columns = {0 : "Height in Inches"})

# Inserting the series
bmidata.insert(3, column = 'Height in Inches', value = heightseries)
```

We can now calculate our desired BMI value. Since the measurements provided are in imperial, we will use the formula:

BMI = (Weight(lb) * 703) / ((Height(inches))^2)

We will also round these values to the first decimal place.

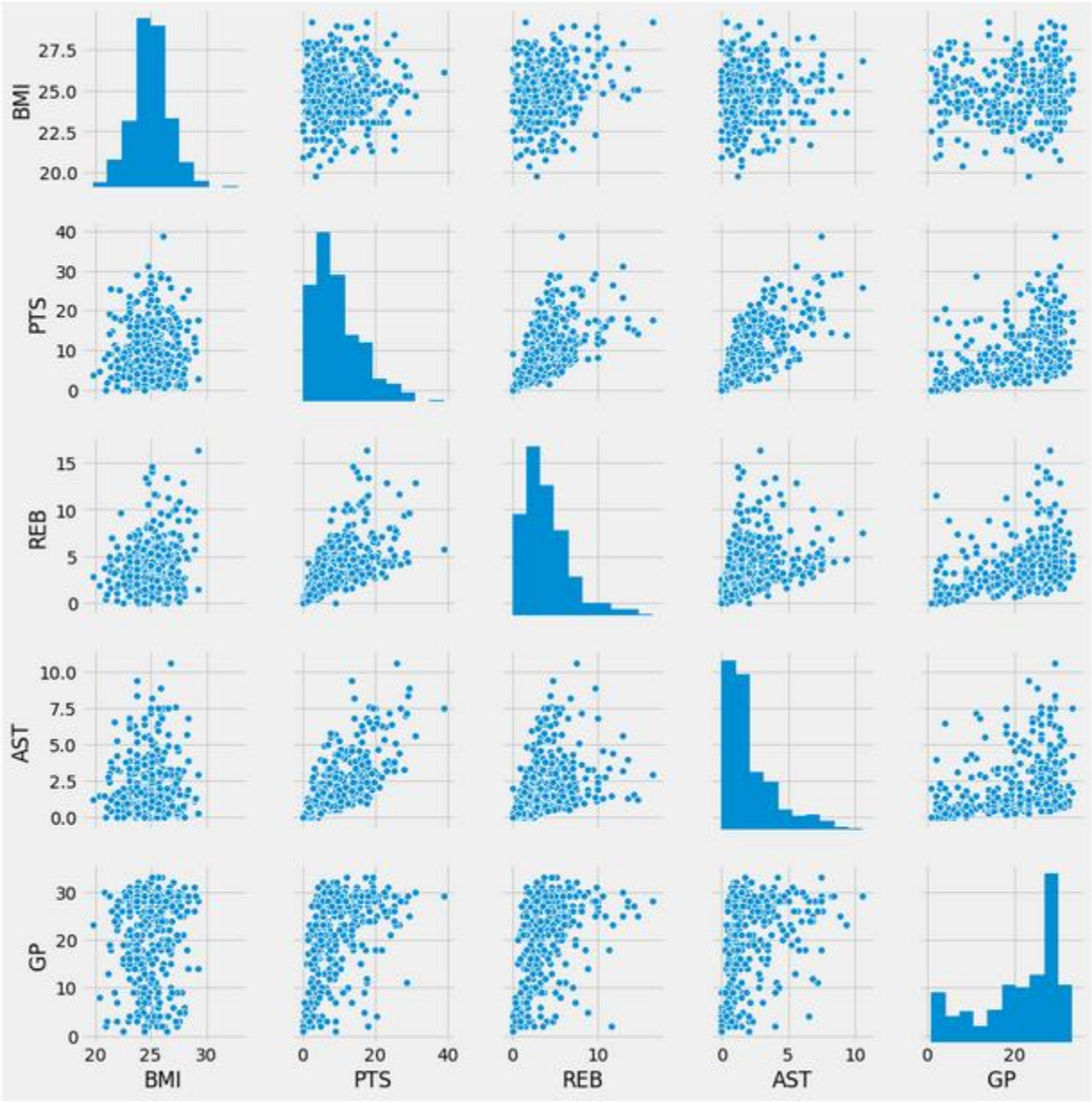
```
# Calculating and inserting BMI column
bmi = ((bmidata['Weight'] * 703) / (bmidata['Height in Inches'] * bmidata['Height in
Inches'])).round(1)
bmi = pd.DataFrame(bmi)
bmidata.insert(6, column = 'BMI', value = bmi)

bmidata.head()
```

	Player	Team	Age	Height in Inches	Height	Weight	BMI	College	Country	Draft Year	Draft Round	Draft Number	GP	PTS	REB	AST	NetRtg
0	Tacko Fall	BOS	24.0	89.0	7-5	311.0	27.6	None	Senegal	Undrafted	Undrafted	Undrafted	3.0	4.3	2.3	0.0	-1.6
1	Boban Marjanovic	DAL	31.0	88.0	7-4	290.0	26.3	None	Serbia	Undrafted	Undrafted	Undrafted	13.0	5.8	4.5	0.5	26.8
2	Zion Williamson	NOP	19.0	78.0	6-6	284.0	32.8	Duke	USA	2019	1	1	NaN	NaN	NaN	NaN	NaN
3	Nikola Jokic	DEN	24.0	84.0	7-0	284.0	28.3	None	Serbia	2014	2	41	28.0	17.3	10.0	6.8	8.2
4	Brook Lopez	MIL	31.0	84.0	7-0	282.0	28.1	Stanford	USA	2008	1	10	29.0	10.0	5.1	1.5	13.4

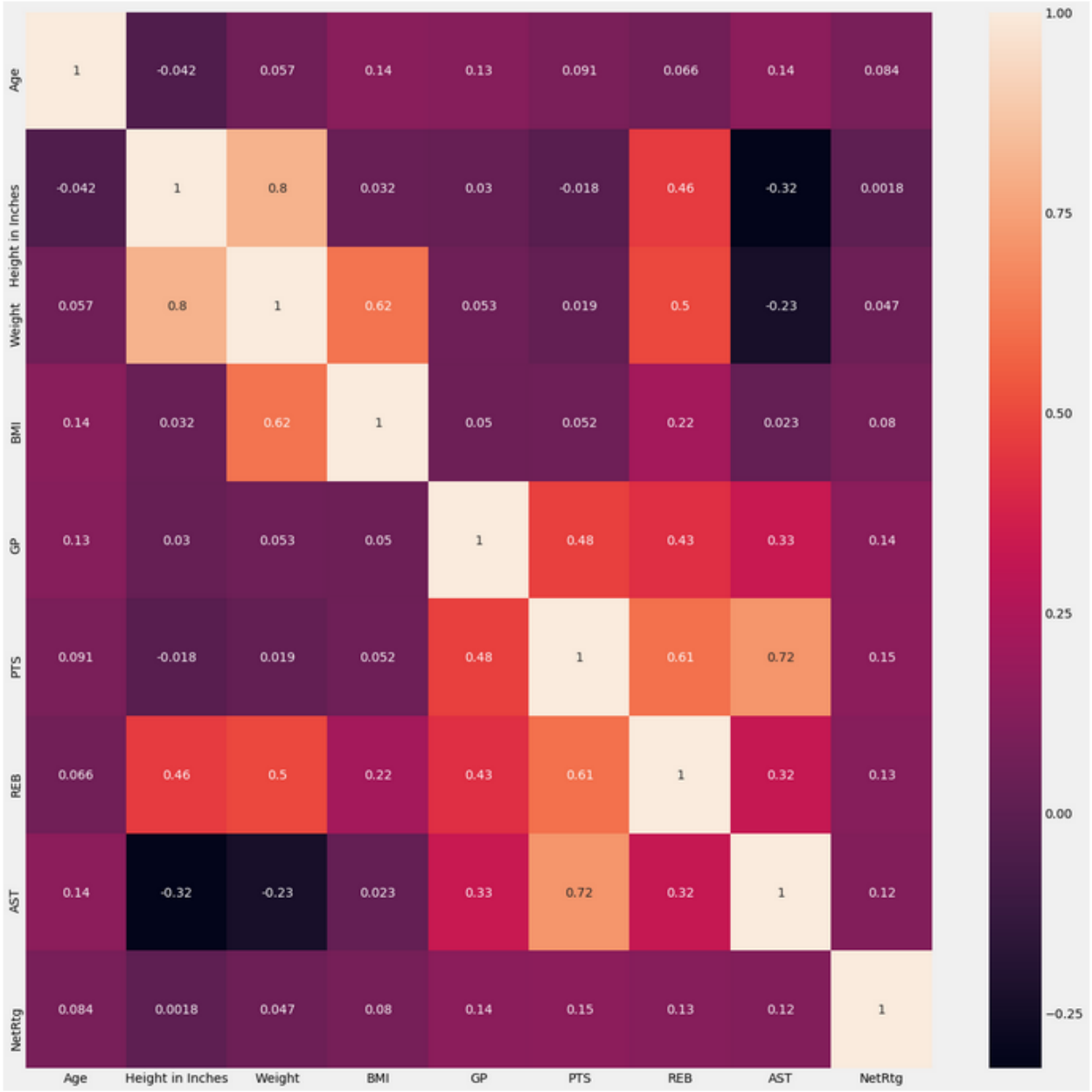
3. Analyzing BMI

Now that we have calculated the BMI values, let's explore the data a little bit before we delve into the topic of Zion Williamson.

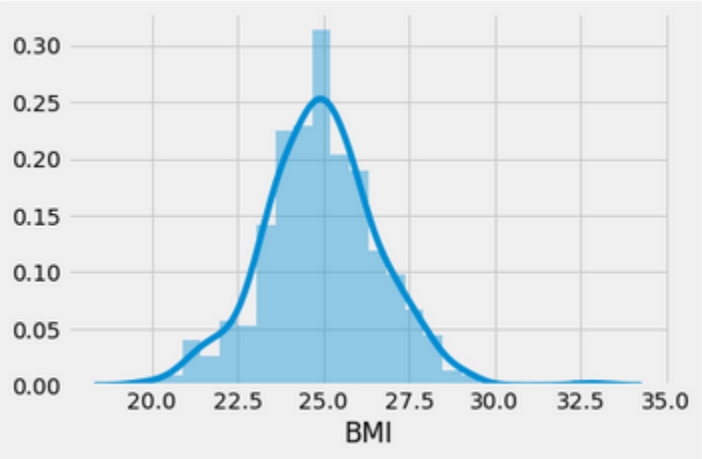
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

sns.pairplot(bmidata[["BMI", "PTS", "REB", "AST", "GP"]])
plt.show()
```



```
plt.figure(figsize = (20,20))
ax = sns.heatmap(bmidata.corr(), annot=True)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```



```
sns.distplot(bmidata['BMI'])
```



So what have we really learned so far? The first plot we generated was a pairplot between BMI, points, rebounds, assists and games played. Interestingly, the plots that compare BMI to the other statistics seem to follow some general form of a normal distribution, except for the number of games played. What we can hypothesize from this is that the BMI of a player does not have a real significant impact on these statistics (meaning a low correlation). This is further evidenced in our correlation heat map, values of interest are as follows:

Correlation between BMI and points: 0.052
Correlation between BMI and rebounds: 0.22
Correlation between BMI and assists: 0.023
Correlation between BMI and games played: 0.14

We notice that the correlation between BMI and rebounds is actually a bit higher than the other correlation values. How can we explain this? Notice that the correlation between BMI and weight is 0.62, which is relatively high compared to other values in the heat map. Investigating further, the correlation between weight and height is significantly high at 0.8. We can hypothesize that there is a cluster of tall players with high BMIs. Since height is a significant factor in getting rebounds (taller players have an easier time getting the ball when bouncing off of the rim), the slight bump in correlation between BMI and rebounds may be explained by this. Although this relationship is still relatively weak, it is a noticeable observation in the correlation values.

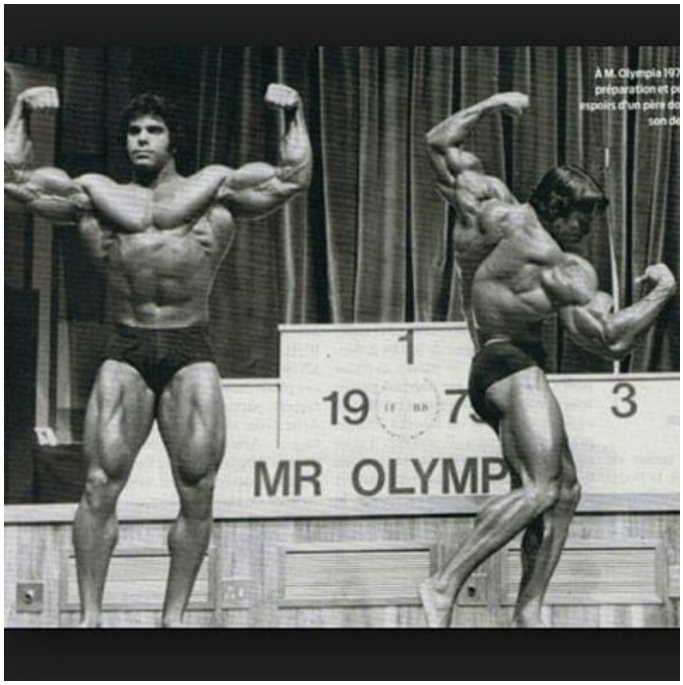
The histogram plot of the distribution of BMI values shows us that it follows a normal distribution with the highest number of players having a BMI value of about 25. Having the data follow a normal distribution is significant because it allows us to perform t tests and z tests on our BMI data. An interesting observation is that a BMI value of 25 is technically considered overweight, although we would not consider most NBA players to be overweight. This is an example of where BMI fails slightly in being an indicator of health. We would consider NBA players, who are by definition professional athletes, to be the pinnacle of health and performance, but they are considered overweight when looked at strictly through the lens on BMI scores.

4. Zion "Gumbotron" Williamson

Zion Williamson, as stated earlier, is predicted to make waves in the NBA. Unfortunately for the 2019 – 2020 season, the only waves he is currently making are the ones in the swimming pool when he jumps in as he is currently **sitting out for the entire season due to injury in the preseason**. On October 13th, 2019, Zion tore his meniscus in a game against the San Antonio Spurs. Talk about a bummer.

There is mixed talk surrounding Zion. On one hand, there are many level headed individuals talking about his weight and how it is detrimental to his physical health and durability in the game. Sitting at 6'6" and 285lb, Zion sits at a BMI of approximately 33 and is considered obese by the BMI chart. Furthermore, a score of 35 would be considered extremely obese and Zion teeters on that wall. Individuals who share this opinion recommend that Zion lose a significant amount of weight for the sake of his health and longevity in the game. There are many reasons why too much weight is bad. One example would be that excess weight bears an unnecessary amount of load on the joints, particularly the knees, and is a contributing factor to injury.

On the other hand, there are some (individuals who make extraordinary claims about Zion. One such claim is that Zion is a freak of nature that packs on inhuman amounts of muscle and is composed of almost 100% muscle. This myth is propogated by the Pelicans management team. This claim is incorrect. Let's compare Zion to someone similar, 6'4" and 275lb. This individual is none other than Lou Ferrigno, a legendary bodybuilder who competed against Arnold in his prime days. The stats listed were Lou Ferrigno's height and competition weight in the 1975 Mr. Olympia. He looked like this:

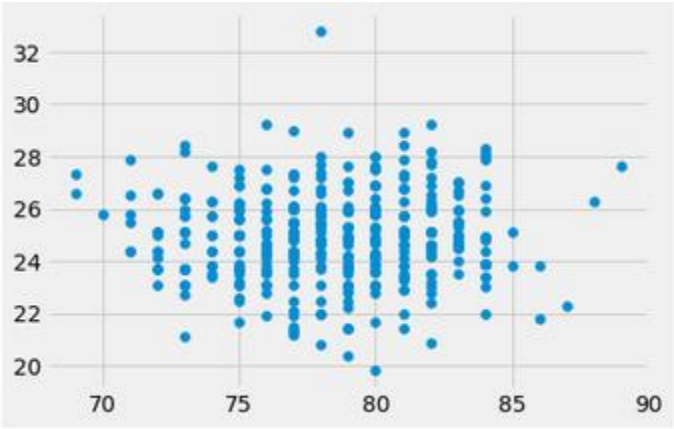


On the left is Lou Ferrigno and on the right is none other than Arnold.

For comparison, here is Zion Williamson:

Although the argument should end here, this paper is about numbers and statistics. So let's run some numbers to look at Zion Williamson.

```
plt.scatter(x = bmidata['Height in Inches'], y = bmidata['BMI'])
```



Above is scatter plot of height vs. BMI in the NBA. The one y-axis outlier we observe, at a BMI score of 33, is none other than Zion Williamson. Notice how Zion's height is almost perfectly in the median of NBA player heights yet his BMI casts a shadow on all other BMI values. He is the Tacko Fall of BMI.

When reading about Zion, many commenters make jokes about how the food in New Orleans is too good to resist and is the reason behind his weight.

By this logic, the Pelicans should have a higher average BMI than other teams as other players would be affected by this factor as well. Let's name this **"The Gumbo Effect"**. To summarize:

The Gumbo Effect is the hypothesis that the food in New Orleans is too good to resist and that people located there will be heavier due to this fact.

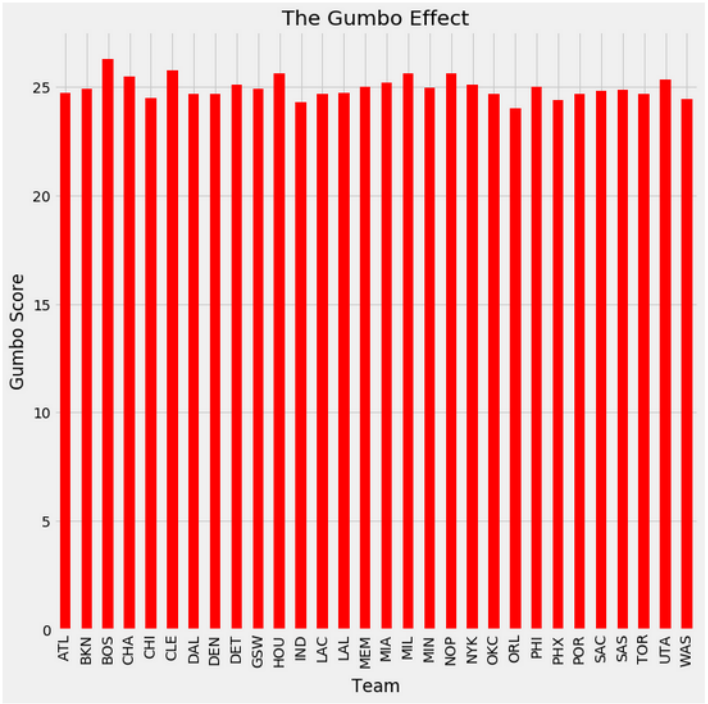
Let's use hypothesis testing to look determine whether or not this is true or false.

H_0: (Average BMI of NOP) – (Average BMI of rest of NBA) == 0 ($\mu1=\mu2$)
H_1: (Average BMI of NOP) – (Average BMI of rest of NBA) =/= 0 ($\mu1\neq\mu2$)

Using Python and Pandas, let's calculate these values. We can then use statsmodels to test.

```
import statsmodels as sm
import statistics

teamsbmi = bmidata.groupby('Team')
teams = ['ATL', 'BKN', 'BOS', 'CHA', 'CHI', 'CLE', 'DAL', 'DEN', 'DET', 'GSW', 'HOU', 'IND',
'LAC', 'LAL', 'MEM', 'MIA', 'MIL', 'MIN', 'NOP', 'NYK', 'OKC', 'ORL', 'PHI', 'PHX', 'POR', 'SAC',
'SAS', 'TOR', 'UTA', 'WAS']

# There are thirty teams in the NBA, let's create a separate dataframe object for all of them
ATL = teamsbmi.get_group('ATL')
BKN = teamsbmi.get_group('BKN')
BOS = teamsbmi.get_group('BOS')
CHA = teamsbmi.get_group('CHA')
CHI = teamsbmi.get_group('CHI')
CLE = teamsbmi.get_group('CLE')
DAL = teamsbmi.get_group('DAL')
DEN = teamsbmi.get_group('DEN')
DET = teamsbmi.get_group('DET')
GSW = teamsbmi.get_group('GSW')
HOU = teamsbmi.get_group('HOU')
IND = teamsbmi.get_group('IND')
LAC = teamsbmi.get_group('LAC')
LAL = teamsbmi.get_group('LAL')
MEM = teamsbmi.get_group('MEM')
MIA = teamsbmi.get_group('MIA')
MIL = teamsbmi.get_group('MIL')
MIN = teamsbmi.get_group('MIN')
NOP = teamsbmi.get_group('NOP')
NYK = teamsbmi.get_group('NYK')
OKC = teamsbmi.get_group('OKC')
ORL = teamsbmi.get_group('ORL')
PHI = teamsbmi.get_group('PHI')
PHX = teamsbmi.get_group('PHX')
POR = teamsbmi.get_group('POR')
SAC = teamsbmi.get_group('SAC')
SAS = teamsbmi.get_group('SAS')
TOR = teamsbmi.get_group('TOR')
UTA = teamsbmi.get_group('UTA')
WAS = teamsbmi.get_group('WAS')
```

```
# Let's create a list of mean BMIs for all teams
meanbmis = [ATL.agg({'BMI' : 'mean'})[0],
                        BKN.agg({'BMI' : 'mean'})[0],
                        BOS.agg({'BMI' : 'mean'})[0],
                        CHA.agg({'BMI' : 'mean'})[0],
                        CHI.agg({'BMI' : 'mean'})[0],
                        CLE.agg({'BMI' : 'mean'})[0],
                        DAL.agg({'BMI' : 'mean'})[0],
                        DEN.agg({'BMI' : 'mean'})[0],
                        DET.agg({'BMI' : 'mean'})[0],
                        GSW.agg({'BMI' : 'mean'})[0],
                        HOU.agg({'BMI' : 'mean'})[0],
                        IND.agg({'BMI' : 'mean'})[0],
                        LAC.agg({'BMI' : 'mean'})[0],
                        LAL.agg({'BMI' : 'mean'})[0],
                        MEM.agg({'BMI' : 'mean'})[0],
                        MIA.agg({'BMI' : 'mean'})[0],
                        MIL.agg({'BMI' : 'mean'})[0],
                        MIN.agg({'BMI' : 'mean'})[0],
                        NOP.agg({'BMI' : 'mean'})[0],
                        NYK.agg({'BMI' : 'mean'})[0],
                        OKC.agg({'BMI' : 'mean'})[0],
                        ORL.agg({'BMI' : 'mean'})[0],
                        PHI.agg({'BMI' : 'mean'})[0],
                        PHX.agg({'BMI' : 'mean'})[0],
                        POR.agg({'BMI' : 'mean'})[0],
                        SAC.agg({'BMI' : 'mean'})[0],
                        SAS.agg({'BMI' : 'mean'})[0],
                        TOR.agg({'BMI' : 'mean'})[0],
                        UTA.agg({'BMI' : 'mean'})[0],
                        WAS.agg({'BMI' : 'mean'})[0]]

meanbmibarplotdf = pd.DataFrame(index = teams, columns = ['Mean BMI'])
meanbmibarplotdf['Mean BMI'] = meanbmis

plt.style.use('fivethirtyeight')
meanbmiplot = meanbmibarplotdf.plot.bar(figsize = (10, 10), color = 'red', legend = False)
plt.title('The Gumbo Effect')
plt.ylabel('Gumbo Score')
plt.xlabel('Team', labelpad = 10)
```



Looking at the bar plot, there seems to be no discernable difference between the average BMI of the NOP vs. all other teams. In fact, there exists teams that have a higher average BMI than the NOP.

If we view all players included in the bmidata dataframe as our population, the group of interest we are looking at are players in the NOP, which has a size of 13 players. We can compare these 13 players to a random sample of 13 other players from other teams to use hypothesis testing to either accept or reject our null hypothesis. Since the sample size is small (n < 30), we use the t-test as the method to reject or accept our null hypothesis. Let's create the first population (players in the NOP) and our random sample (13 other players selected at random).

```
# Creating array with all BMI values from NOP
a = np.array(NOP['BMI'])

# Creating array with all other BMI values
teamsexceptnop = [ATL, BKN, BOS, CHA, CHI, CLE, DAL, DEN, DET, GSW, HOU, IND, LAC, LAL, MEM, MIA,
MIL, MIN, NYK, OKC, ORL, PHI, PHX, POR, SAC, SAS, TOR, UTA, WAS]

otherbmi = np.empty((0, 0))

for i in teamsexceptnop:
   otherbmi = np.append(otherbmi, i['BMI'])

# Creating array with random sample of size N = 13
b = np.random.choice(otherbmi, 13)

# Running the t-test using scipy
t, p = scipy.stats.ttest_ind(a,b)

t
> 1.2927314374559669

p
> 0.20841121777900132
```

We observe that we get a t-score of ~1.29 and a p-value of ~0.21 for this trial run. So p > 0.05 and we cannot reject our null hypothesis. Let's run this test multiple times, each time with a different random sample so we can be more confident in out results.

```
index = 0
results = np.empty((0, 0))
while index < 500:
    b = np.random.choice(otherbmi, 13)
    t, p = scipy.stats.ttest_ind(a,b)
    results = np.append(results, p)
    index = index + 1

results.mean()
> 0.4767516086274639
```

After running this t-test over 500 random samples, it seems to converge around a value of around ~0.48. Since it converges at a value higher than 0.05, we can more confidently say that the null hypothesis cannot be rejected. By this result, the Gumbo Effect does not exist. Players on the New Orleans Pelicans are not actually higher in BMI than the rest of the NBA and Zion Williamson is truly an outlier.

5. More Fun With BMI

All joking aside, Zion Williamson is a very skilled player and may he have all the best in his professional basketball career. Let's take a few more looks at BMI to see if we can identify any patterns. We can start with a PCA analysis to reduce the data down to two dimensions. We can then plot the data by the first two principal components to see if we can identify any clusters. Before we can do this, let's turn our percentage values to floats. We also use pd.dropna() to drop Zion Williamson as he has NaN values.

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

bmiclean = bmidata.dropna()

def p2f(x):
    return float(x.strip('%'))/100

bmiclean['OREB%'] = bmiclean['DREB%'].apply(p2f)
bmiclean['DREB%'] = bmiclean['DREB%'].apply(p2f)
bmiclean['USG%'] = bmiclean['USG%'].apply(p2f)
bmiclean['TS%'] = bmiclean['TS%'].apply(p2f)
bmiclean['AST%'] = bmiclean['AST%'].apply(p2f)
```

We can now start data preprocessing in preparation for our machine learning algorithm. We create X and y. X will contain our independent variables (numerical statistics other than BMI) and y will be our dependent variable (BMI). We then scale X.

```
# Creating X and y
X = bmiclean.iloc[:, np.r_[2:4, 5:7, 12:22]]
y = bmiclean.iloc[:, 6]

# Scaling
X = StandardScaler().fit_transform(X)
```

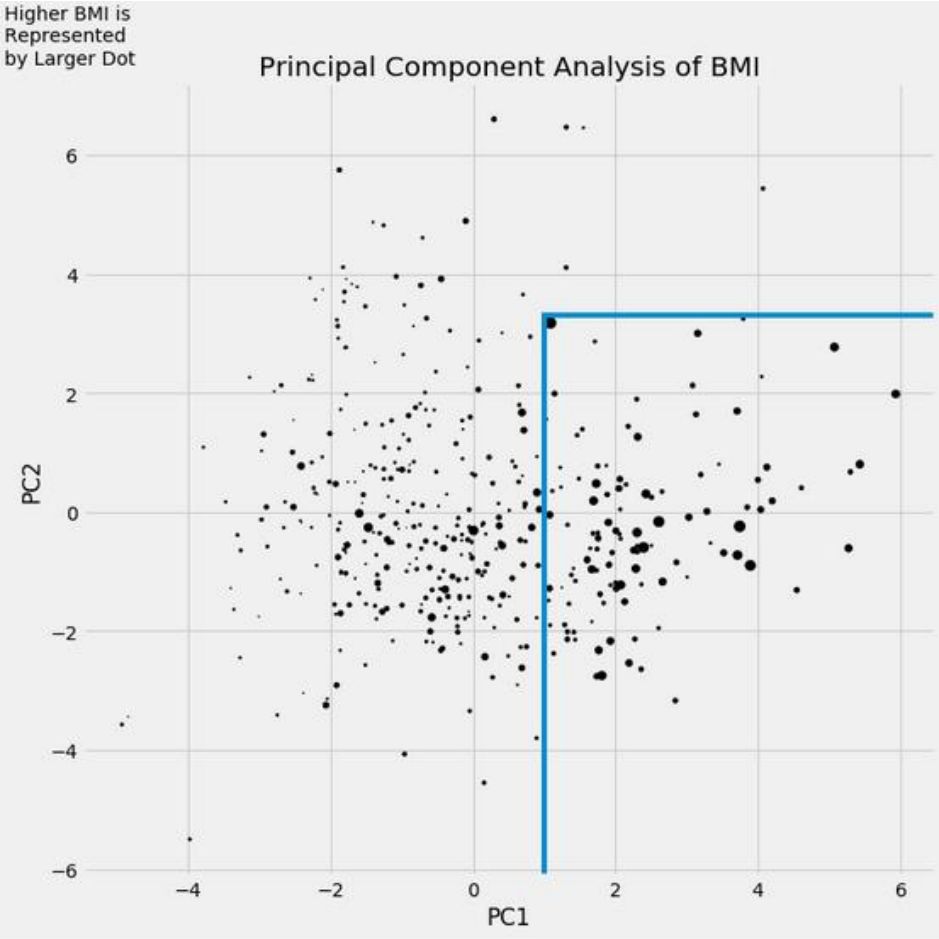
We now construct our first two principal components and create our new DataFrame to be plotted.

```
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X)
principalDf = pd.DataFrame(data = principalComponents ,columns = ['PC1', 'PC2'])
principalDf['BMI'] = y

principaldf.head()
```

	PC1	PC2	BMI
0	5.269294	-0.608310	27.6
1	4.543426	-1.310703	26.3
2	3.602304	3.344854	NaN
3	2.300066	-0.344047	28.3
4	2.071136	-1.224699	28.1

Now that we have our first two principal components, PC1 and PC2, we can create our scatterplot and see if we can identify any clusters. We will make the size of our points dependent on the value of BMI. Higher BMI values will yield larger points

```
plt.figure(figsize = (10, 10))
plt.style.use("fivethirtyeight")
plt.scatter(x = principalDf['PC1'], y = principalDf['PC2'], s =
((principalDf['BMI']/principalDf['BMI'].mean())+0.15) ** 15, c = 'black')
plt.title('Principal Component Analysis of BMI')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.axvline(x = 1, ymax = 0.71)
plt.axhline(y = 3.3, xmin = 0.54)
plt.figtext(x = 0, y = 0.9, s = "Higher BMI is \nRepresented \nby Larger Dot")
```



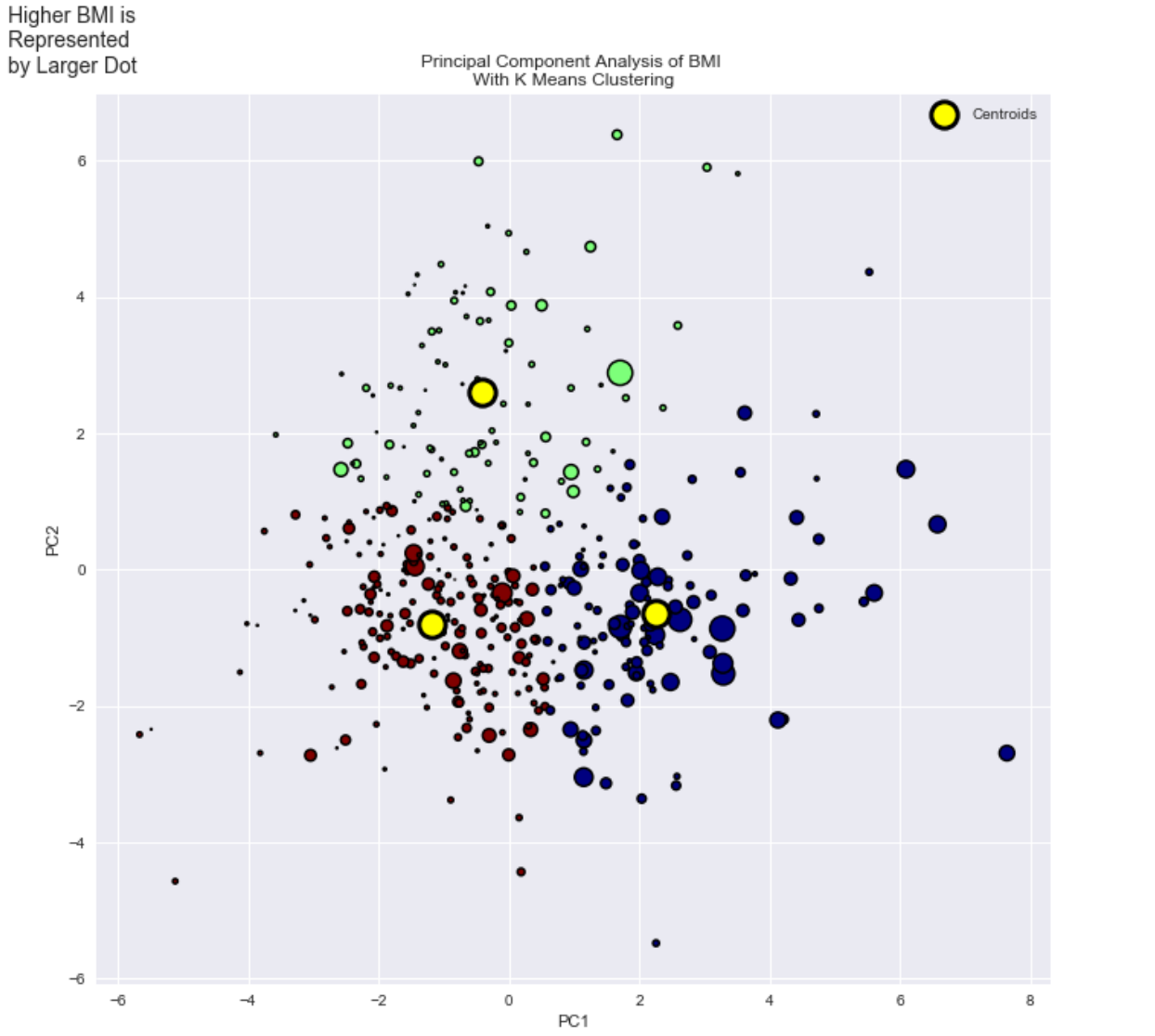
It seems as though we have an interesting result. There looks to be one discernable cluster in our plot where our larger points (higher BMIs) seem to gather. It looks to be the area of the plot bounded by:

$1 < PC1 < 6$
$-6 < PC2 < 3.3$

Could this be the group of high BMI players that produce a high number of rebounds? By the correlation plots produced earlier, it would seem to make sense.

Let's test our clustering theory. With unlabeled data, we can use K Means Clustering to algorithmically group these points into clusters.

```
# K Means Clustering on PCA1 and PCA2
from sklearn.cluster import KMeans
X = principalDf.iloc[:,0:2]

# Applying kmeans to the data
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state =
0)
y_kmeans = kmeans.fit_predict(X)

# PCA1 and PC2 scatter plot with K Means Clustering
plt.figure(figsize = (10, 10))
plt.style.use("seaborn")
plt.scatter(x = principalDf['PC1'], y = principalDf['PC2'], s =
((principalDf['BMI']/principalDf['BMI'].mean())+0.15) ** 20, c=y_kmeans, cmap = 'jet', linewidths
= 1.5, edgecolor = 'black' )
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow',
label = 'Centroids', linewidths = 3, edgecolors = 'black')
plt.title('Principal Component Analysis of BMI \nWith K Means Clustering')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.figtext(x = 0, y = 0.9, s = "Higher BMI is \nRepresented \nby Larger Dot")
plt.legend()
```



Running the K Means algorithm with 3 clusters showed us the cluster which was visually identifiable (blue), a second cluster that is densely populated (red) and a third cluster that is sparsely populated (light green). The three centroids are represented by the yellow dots. Further analysis can tell us more information. This can be achieved by looking at which players belong to each cluster. From there we can look for similar characteristics between those players, etc.